aws

Guide de l'utilisateur d'Amazon EMR Serverless

Amazon EMR

Amazon EMR: Guide de l'utilisateur d'Amazon EMR Serverless

Table of Contents

Qu'est-ce qu'Amazon EMR Serverless ?

Amazon EMR Serverless est une option de déploiement pour Amazon EMR qui fournit un environnement d'exécution sans serveur. Cela simplifie le fonctionnement des applications d'analyse qui utilisent les derniers frameworks open source, tels qu'Apache Spark et Apache Hive. Avec EMR Serverless, vous n'avez pas besoin de configurer, d'optimiser, de sécuriser ou d'exploiter des clusters pour exécuter des applications avec ces frameworks.

EMR Serverless vous permet d'éviter le surprovisionnement ou le sous-provisionnement des ressources pour vos tâches de traitement des données. EMR Serverless détermine automatiquement les ressources dont l'application a besoin, obtient ces ressources pour traiter vos tâches et libère les ressources une fois les tâches terminées. Pour les cas d'utilisation où les applications ont besoin d'une réponse en quelques secondes, comme l'analyse interactive des données, vous pouvez pré-initialiser les ressources dont l'application a besoin lorsque vous créez l'application.

Avec EMR Serverless, vous continuez à bénéficier des avantages d'Amazon EMR, tels que la compatibilité open source, la simultanéité et les performances d'exécution optimisées pour les frameworks les plus courants.

EMR Serverless convient aux clients qui souhaitent utiliser facilement des applications à l'aide de frameworks open source. Il permet un démarrage rapide des tâches, une gestion automatique des capacités et un contrôle simple des coûts.

Concepts

Dans cette section, nous abordons les termes et concepts EMR Serverless qui apparaissent tout au long de notre guide de l'utilisateur EMR Serverless.

Version de sortie

Une version d'Amazon EMR est un ensemble d'applications open source issues de l'écosystème des mégadonnées. Chaque version inclut différentes applications, composants et fonctionnalités Big Data que vous sélectionnez pour le déploiement et la configuration d'EMR Serverless afin qu'ils puissent exécuter vos applications. Lorsque vous créez une application, spécifiez sa version finale. Choisissez la version d'Amazon EMR et la version du framework open source que vous souhaitez utiliser dans votre application. Pour en savoir plus sur les versions préliminaires, reportez-vous àVersions de lancement d'Amazon EMR Serverless.

Application

Avec EMR Serverless, vous pouvez créer une ou plusieurs applications EMR Serverless qui utilisent des frameworks d'analyse open source. Pour créer une application, spécifiez les attributs suivants :

- La version publiée par Amazon EMR correspondant à la version du framework open source que vous souhaitez utiliser. Pour déterminer la version de votre version, reportez-vous àVersions de lancement d'Amazon EMR Serverless.
- Le runtime spécifique que vous souhaitez que votre application utilise, tel qu'Apache Spark ou Apache Hive.

Après avoir créé une candidature, soumettez-y des tâches de traitement des données ou des demandes interactives.

Chaque application EMR Serverless s'exécute sur un Amazon Virtual Private Cloud (VPC) sécurisé, à l'écart des autres applications. Utilisez également des politiques Gestion des identités et des accès AWS (IAM) pour définir les utilisateurs et les rôles autorisés à accéder à l'application. Vous pouvez également définir des limites pour contrôler et suivre les coûts d'utilisation engagés par l'application.

Envisagez de créer plusieurs applications lorsque vous devez effectuer les opérations suivantes :

- Utilisez différents frameworks open source
- Utilisez différentes versions de frameworks open source pour différents cas d'utilisation
- Effectuer des A/B tests lors de la mise à niveau d'une version à une autre
- Maintenir des environnements logiques distincts pour les scénarios de test et de production
- Fournissez des environnements logiques distincts aux différentes équipes avec des contrôles des coûts et un suivi de l'utilisation indépendants
- Séparez les différentes line-of-business applications

EMR Serverless est un service régional qui simplifie la manière dont les charges de travail sont exécutées dans plusieurs zones de disponibilité d'une région. Pour en savoir plus sur l'utilisation des applications avec EMR Serverless, reportez-vous à. Interagissez avec une application EMR sans serveur et configurez-la

Exécution de tâche

Une exécution de tâche est une demande soumise à une application EMR sans serveur que l'application exécute et suit de manière asynchrone jusqu'à son achèvement. Parmi les tâches, citons une requête HiveQL que vous soumettez à une application Apache Hive ou un script de traitement de données que vous soumettez à PySpark une application Apache Spark. Lorsque vous soumettez une tâche, vous devez spécifier un rôle d'exécution, créé dans IAM, que la tâche utilise pour accéder aux AWS ressources, telles que les objets Amazon S3. Vous pouvez envoyer plusieurs demandes d'exécution de tâches à une application, et chaque exécution de tâche peut utiliser un rôle d'exécution différent pour accéder aux AWS ressources. Une application EMR Serverless commence à exécuter des tâches dès qu'elle les reçoit et exécute plusieurs demandes de tâches simultanément. Pour en savoir plus sur la façon dont EMR Serverless exécute les tâches, reportez-vous à. Exécution de tâches

Workers

Une application EMR sans serveur fait appel en interne à des travailleurs pour exécuter vos charges de travail. La taille par défaut de ces travailleurs dépend de votre type d'application et de la version publiée par Amazon EMR. Lorsque vous planifiez l'exécution d'une tâche, remplacez ces tailles.

Lorsque vous soumettez une tâche, EMR Serverless calcule les ressources dont l'application a besoin pour la tâche et planifie les travailleurs. EMR Serverless décompose vos charges de travail en tâches, télécharge des images, approvisionne et configure les travailleurs, puis les met hors service une fois le travail terminé. EMR Serverless fait automatiquement évoluer les collaborateurs à la hausse ou à la baisse en fonction de la charge de travail et du parallélisme requis à chaque étape du travail. Ce dimensionnement automatique vous évite d'avoir à estimer le nombre de travailleurs dont l'application a besoin pour exécuter vos charges de travail.

Capacité pré-initialisée

EMR Serverless fournit une fonctionnalité de capacité préinitialisée qui permet aux employés de rester initialisés et prêts à réagir en quelques secondes. Cette capacité crée efficacement un bassin de travailleurs chaleureux pour une application. Pour configurer cette fonctionnalité pour chaque application, définissez le `initial-capacity` paramètre d'une application. Lorsque vous configurez une capacité préinitialisée, les tâches peuvent démarrer immédiatement afin que vous puissiez implémenter des applications itératives et des tâches urgentes. Pour en savoir plus sur les travailleurs préinitialisés, reportez-vous à. Configuration d'une application lorsque vous travaillez avec EMR Serverless

EMR Studio

EMR Studio est la console utilisateur qui permet de gérer vos applications EMR Serverless. Si aucun studio EMR n'existe dans votre compte lorsque vous créez votre première application EMR Serverless, nous en créons une automatiquement pour vous. Accédez à EMR Studio depuis la console Amazon EMR ou activez l'accès fédéré depuis votre fournisseur d'identité (IdP) via IAM ou IAM Identity Center. Dans ce cas, les utilisateurs peuvent accéder à Studio et gérer les applications EMR Serverless sans accès direct à la console Amazon EMR. Pour en savoir plus sur le fonctionnement des applications EMR Serverless avec EMR Studio, reportez-vous à et. Création d'une application EMR sans serveur à partir de la console EMR Studio Exécution de tâches depuis la console EMR Studio

Conditions préalables pour démarrer avec EMR Serverless

Cette section décrit les prérequis administratifs pour exécuter EMR Serverless. Il s'agit notamment de la configuration du compte et de la gestion des autorisations.

Rubriques

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d’inscription, vous recevrez un appel téléphonique ou un SMS et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l’utilisateur racine a accès à l’ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l’utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d’un utilisateur doté d’un accès administratif

Une fois que vous vous êtes inscrit à un utilisateur administratif Compte AWS, que vous Utilisateur racine d'un compte AWS l'avez sécurisé AWS IAM Identity Center, que vous l'avez activé et que vous en avez créé un, afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l’aide pour vous connecter en utilisant l’utilisateur racine, consultez Connexion en tant qu’utilisateur racine dans le Guide de l’utilisateur Connexion à AWS .
2. Activez l’authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d’un utilisateur doté d’un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d’ AWS IAM Identity Center dans le Guide de l’utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu’utilisateur doté d’un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l’URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l’utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribution d’un accès à d’autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d’autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

   Pour obtenir des instructions, consultez Création d’un ensemble d’autorisations dans le Guide de l’utilisateur AWS IAM Identity Center .
2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

   Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l’utilisateur AWS IAM Identity Center .

Accorder des autorisations

Dans les environnements de production, nous vous conseillons d'utiliser des politiques plus précises. Pour des exemples de telles politiques, reportez-vous àExemples de politiques d'accès utilisateur pour EMR Serverless. Pour en savoir plus sur la gestion des accès, reportez-vous à la section Gestion des accès pour les AWS ressources du Guide de l'utilisateur IAM.

Pour les utilisateurs qui ont besoin de démarrer avec EMR Serverless dans un environnement sandbox, utilisez une politique similaire à la suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRStudioCreate",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl",
        "elasticmapreduce:DescribeStudio",
```

```
        "elasticmapreduce:CreateStudio",
        "elasticmapreduce:ListStudios"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "EMRServerlessFullAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowEMRServerlessServiceLinkedRoleCreation",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/*"
      ]
    }
  ]
```

```
}
```

Pour activer l’accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d’autorisations. Suivez les instructions de la rubrique Création d’un jeu d’autorisations du Guide de l’utilisateur AWS IAM Identity Center .
- Utilisateurs gérés dans IAM par un fournisseur d’identité :

  Créez un rôle pour la fédération d’identité. Suivez les instructions de la rubrique Création d’un rôle pour un fournisseur d’identité tiers (fédération) dans le Guide de l’utilisateur IAM.
- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d’un rôle pour un utilisateur IAM dans le Guide de l’utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d’utilisateurs. Suivez les instructions de la rubrique Ajout d’autorisations à un utilisateur (console) du Guide de l’utilisateur IAM.

Octroi d’un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	À	Méthode
IAM	(Recommandé) Utilisez les informations d'identification de la console comme informati ons d'identification temporair es pour signer les demandes programmatiques adressées	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Connexion pour le développement AWS local

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
	au AWS CLI AWS SDKs, ou AWS APIs.	dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, voir Connexion pour le développement AWS local dans le guide de référence AWS SDKs and Tools.
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Installez et configurez AWS CLI

Si vous souhaitez utiliser EMR Serverless APIs, installez la dernière version de (). AWS Command Line Interface AWS CLI Vous n'en avez pas besoin AWS CLI pour utiliser EMR Serverless depuis la console EMR Studio et démarrer sans la CLI en suivant les étapes décrites dans. Commencer à utiliser EMR Serverless depuis la console

Pour configurer le AWS CLI

1. Pour installer la dernière version du AWS CLI pour macOS, Linux ou Windows, reportez-vous à la section Installation ou mise à jour de la dernière version du AWS CLI.

2. Pour configurer AWS CLI et sécuriser votre accès à Services AWS, y compris à EMR Serverless, reportez-vous à la section Configuration rapide avec. `aws configure`
3. Pour vérifier la configuration, entrez la DataBrew commande suivante à l'invite de commande.

```
aws emr-serverless help
```

AWS CLI les commandes utilisent la valeur par défaut Région AWS de votre configuration, sauf si vous la définissez avec un paramètre ou un profil. Pour définir Région AWS un paramètre, ajoutez-le à chaque commande. `--region`

Pour définir votre Région AWS profil, ajoutez d'abord un profil nommé dans le `~/.aws/config` fichier ou le `%UserProfile%/.aws/config` fichier (pour Microsoft Windows). Suivez les étapes décrites dans la section Profils nommés pour le AWS CLI. Ensuite, définissez vos paramètres Région AWS et les autres à l'aide d'une commande similaire à celle de l'exemple suivant.

```
[profile emr-serverless]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

Ouvrez la console .

La plupart des rubriques orientées console de cette section partent de la console Amazon EMR.
Si vous n'êtes pas encore connecté à votre Compte AWS, connectez-vous, puis ouvrez la console Amazon EMR et passez à la section suivante pour continuer à démarrer avec Amazon EMR.

Commencer à utiliser Amazon EMR Serverless

Ce didacticiel vous aide à démarrer avec EMR Serverless lorsque vous déployez un exemple de charge de travail Spark ou Hive. Vous allez créer, exécuter et déboguer votre propre application. Nous présentons les options par défaut dans la plupart des parties de ce didacticiel.

Avant de lancer une application EMR sans serveur, effectuez les tâches suivantes.

Rubriques

Accorder des autorisations pour utiliser EMR Serverless

Pour utiliser EMR Serverless, vous avez besoin d'un utilisateur ou d'un rôle IAM associé à une politique qui accorde des autorisations pour EMR Serverless. Pour créer un utilisateur et associer la politique appropriée à cet utilisateur, suivez les instructions figurant dansAccorder des autorisations.

Préparer le stockage pour EMR Serverless

Dans ce didacticiel, vous allez utiliser un compartiment S3 pour stocker les fichiers de sortie et les journaux de l'exemple de charge de travail Spark ou Hive que vous exécuterez à l'aide d'une application EMR sans serveur. Pour créer un bucket, suivez les instructions de la section Création d'un bucket dans le guide de l'utilisateur de la console Amazon Simple Storage Service. Remplacez toute autre référence *amzn-s3-demo-bucket* à par le nom du compartiment nouvellement créé.

Création d'un studio EMR pour exécuter des charges de travail interactives

Si vous souhaitez utiliser EMR Serverless pour exécuter des requêtes interactives via des blocs-notes hébergés dans EMR Studio, vous devez spécifier un compartiment S3 et le rôle de service

minimum pour EMR Serverless afin de créer un espace de travail. Pour connaître les étapes de configuration, consultez la section Configurer un studio EMR dans le guide de gestion Amazon EMR. Pour plus d'informations sur les charges de travail interactives, consultezExécutez des charges de travail interactives avec EMR Serverless via EMR Studio.

Création d'un rôle d'exécution de tâches

Les exécutions de tâches dans EMR Serverless utilisent un rôle d'exécution qui fournit des autorisations détaillées à des ressources spécifiques Services AWS lors de l'exécution. Dans ce didacticiel, un compartiment S3 public héberge les données et les scripts. Le bucket *amzn-s3-demo-bucket* stocke la sortie.

Pour configurer un rôle d'exécution de tâche, créez d'abord un rôle d'exécution avec une politique de confiance afin qu'EMR Serverless puisse utiliser le nouveau rôle. Ensuite, associez la politique d'accès S3 requise à ce rôle. Les étapes suivantes vous guident tout au long du processus.

Console

1. Accédez à la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).
4. La page Créer une politique s'ouvre dans un nouvel onglet. Sélectionnez l'éditeur de politique en tant que Json et collez le JSON de politique ci-dessous.

> **Important**
>
> Remplacez *amzn-s3-demo-bucket* dans la politique ci-dessous par le nom réel du bucket créé dansPréparer le stockage pour EMR Serverless. Il s'agit d'une politique de base pour l'accès à S3. Pour d'autres exemples de rôles d'exécution de tâches, consultezRôles d'exécution des tâches pour Amazon EMR Serverless.

JSON

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

```
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.elasticmapreduce",
        "arn:aws:s3:::*.elasticmapreduce/*"
      ]
    },
    {
      "Sid": "FullAccessToOutputBucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
```

```
      "Resource": [
        "*"
      ]
    }
  ]
}
```

5. Choisissez Suivant pour saisir le nom de votre politique, tel que `EMRServerlessS3AndGlueAccessPolicy` « Créer une politique »
6. Dans le volet de navigation gauche de la console IAM, sélectionnez Rôles.
7. Choisissez Créer un rôle.
8. Pour le type de rôle, choisissez Politique de confiance personnalisée et collez la politique de confiance suivante. Cela permet aux tâches soumises à vos applications Amazon EMR Serverless d'accéder à d'autres applications en votre Services AWS nom.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/
EMRServerlessExecutionRole",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

9. Choisissez Suivant pour accéder à la page Ajouter des autorisations, puis sélectionnez EMRServerlessS3 AndGlueAccessPolicy.
10. Dans la page Nom, révision et création, pour Nom du rôle, entrez le nom de votre rôle, par exemple,`EMRServerlessS3RuntimeRole`. Pour créer ce rôle IAM, choisissez Create role.

CLI

1. Créez un fichier nommé `emr-serverless-trust-policy.json` contenant la stratégie d'approbation à utiliser pour le rôle IAM. Le fichier doit contenir la politique suivante.

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Sid": "EMRServerlessTrustPolicy",
         "Action": [
           "sts:AssumeRole"
         ],
         "Effect": "Allow",
         "Resource": "arn:aws:iam::123456789012:role/
   EMRServerlessExecutionRole"
       }
     ]
   }
   ```

2. Créez un rôle IAM nommé `EMRServerlessS3RuntimeRole`. Utilisez la politique de confiance que vous avez créée à l'étape précédente.

   ```
   aws iam create-role \
       --role-name EMRServerlessS3RuntimeRole \
       --assume-role-policy-document file://emr-serverless-trust-policy.json
   ```

   Notez l'ARN dans la sortie. Vous utilisez l'ARN du nouveau rôle lors de la soumission des tâches, appelé ensuite*`job-role-arn`*.

3. Créez un fichier nommé `emr-sample-access-policy.json` qui définit la politique IAM pour votre charge de travail. Cela fournit un accès en lecture au script et aux données stockées dans des compartiments S3 publics et un accès en lecture-écriture à. *`amzn-s3-demo-bucket`*

   > **Important**
   >
   > Remplacez *`amzn-s3-demo-bucket`* dans la politique ci-dessous par le nom réel du bucket créé dansPréparer le stockage pour EMR Serverless.. Il s'agit d'une politique

de base pour l'accès à AWS Glue et S3. Pour d'autres exemples de rôles d'exécution de tâches, consultezRôles d'exécution des tâches pour Amazon EMR Serverless.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.elasticmapreduce",
        "arn:aws:s3:::*.elasticmapreduce/*"
      ]
    },
    {
      "Sid": "FullAccessToOutputBucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
```

```
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

4. Créez une stratégie IAM nommée `EMRServerlessS3AndGlueAccessPolicy` avec le fichier de stratégie que vous avez créé à l'étape 3. Prenez note de l'ARN dans la sortie, car vous utiliserez l'ARN de la nouvelle politique à l'étape suivante.

   ```
   aws iam create-policy \
       --policy-name EMRServerlessS3AndGlueAccessPolicy \
       --policy-document file://emr-sample-access-policy.json
   ```

   Notez l'ARN de la nouvelle politique dans le résultat. Vous le remplacerez *`policy-arn`* à l'étape suivante.

5. Associez la politique IAM `EMRServerlessS3AndGlueAccessPolicy` au rôle `EMRServerlessS3RuntimeRole` d'exécution du job.

   ```
   aws iam attach-role-policy \
       --role-name EMRServerlessS3RuntimeRole \
       --policy-arn policy-arn
   ```

Commencer à utiliser EMR Serverless depuis la console

Cette section décrit l'utilisation d'EMR Serverless, notamment la création d'un studio EMR. Il décrit également comment soumettre des exécutions de tâches et consulter les journaux.

Étapes à suivre

Étape 1 : Création d'une application EMR sans serveur

Créez une nouvelle application avec EMR Serverless comme suit.

1. Connectez-vous à la console Amazon EMR AWS Management Console et ouvrez-la à https://console.aws.amazon.com l'adresse /emr.
2. Dans le volet de navigation de gauche, choisissez EMR Serverless pour accéder à la page d'accueil EMR Serverless.
3. Pour créer ou gérer des applications EMR Serverless, vous avez besoin de l'interface utilisateur d'EMR Studio.

   - Si vous disposez déjà d'un studio EMR dans Région AWS lequel vous souhaitez créer une application, sélectionnez Gérer les applications pour accéder à votre studio EMR ou sélectionnez le studio que vous souhaitez utiliser.
   - Si vous ne disposez pas d'un studio EMR dans l' Région AWS endroit où vous souhaitez créer une application, choisissez Commencer, puis Choisissez Créer et lancez Studio. EMR Serverless crée pour vous un studio EMR afin que vous puissiez créer et gérer des applications.
4. Dans l'interface utilisateur de Create studio qui s'ouvre dans un nouvel onglet, entrez le nom, le type et la version de publication de votre application. Si vous souhaitez uniquement exécuter des tâches par lots, sélectionnez Utiliser les paramètres par défaut pour les tâches par lots uniquement. Pour les charges de travail interactives, sélectionnez Utiliser les paramètres par défaut pour les charges de travail interactives. Vous pouvez également exécuter des tâches par lots sur des applications interactives grâce à cette option. Si nécessaire, vous pouvez modifier ces paramètres ultérieurement.

   Pour plus d'informations, voir Création d'un studio.
5. Sélectionnez Créer une application pour créer votre première application.

Passez à la section suivante Étape 2 : Soumettre une exécution de tâche ou une charge de travail interactive pour soumettre une exécution de tâche ou une charge de travail interactive.

Étape 2 : Soumettre une exécution de tâche ou une charge de travail interactive

Spark job run

Dans ce didacticiel, nous utilisons un PySpark script pour calculer le nombre d'occurrences de mots uniques dans plusieurs fichiers texte. Un compartiment S3 public en lecture seule stocke à la fois le script et le jeu de données.

Pour exécuter une tâche Spark

1. Téléchargez l'exemple de script `wordcount.py` dans votre nouveau compartiment à l'aide de la commande suivante.

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://amzn-s3-demo-bucket/scripts/
```

2. Si vous Étape 1 : Création d'une application EMR sans serveur le complétez, vous accédez à la page des détails de l'application dans EMR Studio. À cet endroit, choisissez l'option Soumettre le travail.
3. Sur la page Soumettre une tâche, effectuez les opérations suivantes.

   - Dans le champ Nom, entrez le nom que vous souhaitez appeler votre tâche exécutée.
   - Dans le champ Rôle d'exécution, entrez le nom du rôle que vous avez créé dansCréation d'un rôle d'exécution de tâches.
   - Dans le champ Emplacement du script, entrez `s3://amzn-s3-demo-bucket/scripts/wordcount.py` l'URI S3.
   - Dans le champ Arguments du script, entrez`["s3://amzn-s3-demo-bucket/emr-serverless-spark/output"]`.
   - Dans la section des propriétés de Spark, choisissez Modifier sous forme de texte et entrez les configurations suivantes.

```
--conf spark.executor.cores=1 --conf spark.executor.memory=4g --
conf spark.driver.cores=1 --conf spark.driver.memory=4g --conf
 spark.executor.instances=1
```

4. Pour démarrer l'exécution de la tâche, choisissez Soumettre la tâche.
5. Dans l'onglet Exécutions de tâches, vous devriez voir votre nouvelle tâche exécutée avec le statut En cours d'exécution.

Hive job run

Dans cette partie du didacticiel, nous créons une table, insérons quelques enregistrements et exécutons une requête d'agrégation de comptes. Pour exécuter la tâche Hive, créez d'abord un fichier contenant toutes les requêtes Hive à exécuter dans le cadre d'une tâche unique, téléchargez le fichier sur S3 et spécifiez ce chemin S3 au démarrage de la tâche Hive.

Pour exécuter une tâche Hive

1. Créez un fichier appelé `hive-query.ql` contenant toutes les requêtes que vous souhaitez exécuter dans votre tâche Hive.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. `hive-query.ql`Téléchargez-le dans votre compartiment S3 à l'aide de la commande suivante.

```
aws s3 cp hive-query.ql s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql
```

3. Si vous Étape 1 : Création d'une application EMR sans serveur le complétez, vous accédez à la page des détails de l'application dans EMR Studio. À cet endroit, choisissez l'option Soumettre le travail.
4. Sur la page Soumettre une tâche, effectuez les opérations suivantes.
   - Dans le champ Nom, entrez le nom que vous souhaitez appeler votre tâche exécutée.
   - Dans le champ Rôle d'exécution, entrez le nom du rôle que vous avez créé dansCréation d'un rôle d'exécution de tâches.
   - Dans le champ Emplacement du script, entrez `s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql` l'URI S3.

- Dans la section Propriétés de la ruche, choisissez Modifier sous forme de texte et entrez les configurations suivantes.

```
--hiveconf hive.log.explain.output=false
```

- Dans la section Configuration du Job, choisissez Modifier au format JSON, puis entrez le code JSON suivant.

```
{
   "applicationConfiguration":
   [{
       "classification": "hive-site",
          "properties": {
              "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/scratch",
              "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
              "hive.driver.cores": "2",
              "hive.driver.memory": "4g",
              "hive.tez.container.size": "4096",
              "hive.tez.cpu.vcores": "1"
           }
    }]
}
```

5. Pour démarrer l'exécution de la tâche, choisissez Soumettre la tâche.
6. Dans l'onglet Exécutions de tâches, vous devriez voir votre nouvelle tâche exécutée avec le statut En cours d'exécution.

Interactive workload

Avec Amazon EMR 6.14.0 et versions ultérieures, vous pouvez utiliser des blocs-notes hébergés dans EMR Studio pour exécuter des charges de travail interactives pour Spark dans EMR Serverless. Pour plus d'informations, notamment sur les autorisations et les conditions requises, consultezExécutez des charges de travail interactives avec EMR Serverless via EMR Studio.

Une fois que vous avez créé votre application et configuré les autorisations requises, procédez comme suit pour exécuter un bloc-notes interactif avec EMR Studio :

1. Accédez à l'onglet Espaces de travail dans EMR Studio. Si vous devez toujours configurer un emplacement de stockage Amazon S3 et un rôle de service EMR Studio, sélectionnez le bouton Configurer le studio dans la bannière en haut de l'écran.
2. Pour accéder à un bloc-notes, sélectionnez un espace de travail ou créez-en un nouveau. Utilisez le lancement rapide pour ouvrir votre espace de travail dans un nouvel onglet.
3. Accédez à l'onglet qui vient d'être ouvert. Sélectionnez l'icône Calculer dans le menu de navigation de gauche. Sélectionnez EMR Serverless comme type de calcul.
4. Sélectionnez l'application interactive que vous avez créée dans la section précédente.
5. Dans le champ Rôle d'exécution, entrez le nom du rôle IAM que votre application EMR Serverless peut assumer pour l'exécution de la tâche. Pour en savoir plus sur les rôles d'exécution, consultez la section Rôles d'exécution Job dans le guide de l'utilisateur Amazon EMR Serverless.
6. Sélectionnez Joindre. Cela peut prendre jusqu'à une minute. La page sera actualisée une fois jointe.
7. Choisissez un noyau et lancez un bloc-notes. Vous pouvez également consulter des exemples de blocs-notes sur EMR Serverless et les copier dans votre espace de travail. Pour accéder aux exemples de blocs-notes, accédez au **`{...}`**menu de navigation de gauche et parcourez les blocs-notes dont le nom de fichier est indiqué `serverless` dans le bloc-notes.
8. Dans le bloc-notes, vous pouvez accéder au lien du journal du pilote et à un lien vers l'interface utilisateur d'Apache Spark, une interface en temps réel qui fournit des indicateurs pour surveiller votre travail. Pour plus d'informations, consultez la section Surveillance des applications et des tâches EMR sans serveur dans le guide de l'utilisateur Amazon EMR Serverless.

Lorsque vous associez une application à un espace de travail Studio, le démarrage de l'application se déclenche automatiquement si elle n'est pas déjà en cours d'exécution. Vous pouvez également prédémarrer l'application et la garder prête avant de l'associer à l'espace de travail.

Étape 3 : Afficher l'interface utilisateur et les journaux de l'application

Pour afficher l'interface utilisateur de l'application, identifiez d'abord le travail exécuté. Une option pour l'interface utilisateur Spark ou Hive Tez est disponible dans la première ligne d'options pour cette tâche exécutée, en fonction du type de tâche. Sélectionnez l'option appropriée.

Si vous avez choisi l'interface utilisateur Spark, cliquez sur l'onglet Executors pour afficher les journaux des pilotes et des exécuteurs. Si vous avez choisi l'interface utilisateur Hive Tez, cliquez sur l'onglet Toutes les tâches pour afficher les journaux.

Une fois que le statut d'exécution de la tâche indique Success, vous pouvez consulter le résultat de la tâche dans votre compartiment S3.

Étape 4 : nettoyer

Bien que l'application que vous avez créée doive s'arrêter automatiquement après 15 minutes d'inactivité, nous vous recommandons tout de même de libérer des ressources que vous n'avez pas l'intention de réutiliser.

Pour supprimer l'application, accédez à la page Répertorier les applications. Sélectionnez l'application que vous avez créée et choisissez Actions → Arrêter pour arrêter l'application. Une fois que l'application est dans `STOPPED` cet état, sélectionnez-la et choisissez Actions → Supprimer.

Pour d'autres exemples d'exécution de tâches Spark et Hive, consultez Utilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur etUtilisation de configurations Hive lorsque vous exécutez des tâches EMR sans serveur.

Commencer à partir du AWS CLI

Commencez à utiliser EMR Serverless à l' AWS CLI aide des commandes with pour créer une application, exécuter des tâches, vérifier le résultat de l'exécution des tâches et supprimer vos ressources.

Étape 1 : Création d'une application EMR sans serveur

Utilisez la `emr-serverless create-application`commande pour créer votre première application EMR Serverless. Vous devez spécifier le type d'application et l'étiquette de publication Amazon EMR associés à la version de l'application que vous souhaitez utiliser. Le nom de l'application est facultatif.

Spark

Pour créer une application Spark, exécutez la commande suivante.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
    --type "SPARK" \
```

```
    --name my-application
```

Hive

Pour créer une application Hive, exécutez la commande suivante.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
    --type "HIVE" \
    --name my-application
```

Notez l'ID de l'application renvoyé dans la sortie. Vous utiliserez cet identifiant pour démarrer la candidature et lors de la soumission des offres d'emploi, appelé ensuite*`application-id`*.

Avant de passer àÉtape 2 : Soumettre une tâche exécutée à votre application EMR Serverless, assurez-vous que votre application a atteint l'`CREATED`état correspondant à l'`get-application`API.

```
aws emr-serverless get-application \
    --application-id application-id
```

EMR Serverless crée des travailleurs pour répondre aux tâches que vous demandez. Par défaut, ils sont créés à la demande, mais vous pouvez également spécifier une capacité pré-initialisée en définissant le `initialCapacity` paramètre lors de la création de l'application. Vous pouvez également limiter la capacité maximale totale qu'une application peut utiliser avec le `maximumCapacity` paramètre. Pour en savoir plus sur ces options, consultez Configuration d'une application lorsque vous travaillez avec EMR Serverless.

Étape 2 : Soumettre une tâche exécutée à votre application EMR Serverless

Votre application EMR Serverless est maintenant prête à exécuter des tâches.

Spark

Dans cette étape, nous utilisons un PySpark script pour calculer le nombre d'occurrences de mots uniques dans plusieurs fichiers texte. Un compartiment S3 public en lecture seule stocke à la fois le script et le jeu de données. L'application envoie le fichier de sortie et les données du journal depuis le moteur d'exécution Spark vers le compartiment S3 que vous avez créé `/output` et les `/logs` répertoires qu'il contient.

Pour exécuter une tâche Spark

1. Utilisez la commande suivante pour copier le script d'exemple que nous allons exécuter dans votre nouveau compartiment.

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://amzn-s3-demo-bucket/scripts/
```

2. Dans la commande suivante, remplacez-le par *application-id* l'ID de votre application. *job-role-arn*Remplacez-le par l'ARN du rôle d'exécution dans lequel vous l'avez crééCréation d'un rôle d'exécution de tâches. *job-run-name*Remplacez-le par le nom que vous souhaitez appeler votre tâche exécutée. Remplacez toutes les *amzn-s3-demo-bucket* chaînes par le compartiment Amazon S3 que vous avez créé et ajoutez-les `/output` au chemin. Cela crée un nouveau dossier dans votre compartiment dans lequel EMR Serverless peut copier les fichiers de sortie de votre application.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --name job-run-name \
    --job-driver '{
        "sparkSubmit": {
          "entryPoint": "s3://amzn-s3-demo-bucket/scripts/wordcount.py",
          "entryPointArguments": ["s3://amzn-s3-demo-bucket/emr-serverless-
spark/output"],
          "sparkSubmitParameters": "--conf spark.executor.cores=1
 --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
 spark.driver.memory=4g --conf spark.executor.instances=1"
        }
    }'
```

3. Notez l'ID d'exécution de la tâche renvoyé dans la sortie. *job-run-id*Remplacez-le par cet ID dans les étapes suivantes.

Hive

Dans ce didacticiel, nous créons une table, insérons quelques enregistrements et exécutons une requête d'agrégation de comptes. Pour exécuter la tâche Hive, créez d'abord un fichier contenant toutes les requêtes Hive à exécuter dans le cadre d'une tâche unique, téléchargez le fichier sur S3 et spécifiez ce chemin S3 lorsque vous démarrez la tâche Hive.

Pour exécuter une tâche Hive

1. Créez un fichier appelé `hive-query.ql` contenant toutes les requêtes que vous souhaitez exécuter dans votre tâche Hive.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. `hive-query.ql`Téléchargez-le dans votre compartiment S3 à l'aide de la commande suivante.

```
aws s3 cp hive-query.ql s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-
query.ql
```

3. Dans la commande suivante, remplacez-le *`application-id`* par votre propre ID d'application. *`job-role-arn`*Remplacez-le par l'ARN du rôle d'exécution dans lequel vous l'avez créé[Création d'un rôle d'exécution de tâches](). Remplacez toutes les *`amzn-s3-demo-bucket`* chaînes par le compartiment Amazon S3 que vous avez créé, puis ajoutez `/output` et `/logs` au chemin. Cela crée de nouveaux dossiers dans votre compartiment, dans lesquels EMR Serverless peut copier les fichiers de sortie et les fichiers journaux de votre application.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
          "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-
query.ql",
          "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
      "applicationConfiguration": [{
        "classification": "hive-site",
          "properties": {
```

```
            "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-
hive/hive/scratch",
            "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
            "hive.driver.cores": "2",
            "hive.driver.memory": "4g",
            "hive.tez.container.size": "4096",
            "hive.tez.cpu.vcores": "1"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/emr-serverless-hive/logs"
           }
        }
    }'
```

4. Notez l'ID d'exécution de la tâche renvoyé dans la sortie. *job-run-id*Remplacez-le par cet ID dans les étapes suivantes.

Étape 3 : passez en revue le résultat de votre exécution

L'exécution de la tâche doit généralement prendre 3 à 5 minutes.

Spark

Vous pouvez vérifier l'état de votre tâche Spark à l'aide de la commande suivante.

```
aws emr-serverless get-job-run \
    --application-id application-id \
    --job-run-id job-run-id
```

Lorsque la destination de votre journal est définie surs3://*amzn-s3-demo-bucket*/emr-serverless-spark/logs, vous pouvez trouver les journaux de cette tâche spécifique exécutée souss3://*amzn-s3-demo-bucket*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*.

Pour les applications Spark, EMR Serverless envoie les journaux d'événements toutes les 30 secondes vers le sparklogs dossier de destination de votre journal S3. Lorsque votre tâche est terminée, les journaux d'exécution de Spark pour le pilote et les exécuteurs sont téléchargés dans des dossiers nommés de manière appropriée par le type de travailleur, tels que driver

ou`executor`. La sortie de la PySpark tâche est transférée vers. `s3://amzn-s3-demo-bucket/output/`

Hive

Vous pouvez vérifier l'état de votre tâche Hive à l'aide de la commande suivante.

```
aws emr-serverless get-job-run \
    --application-id application-id \
    --job-run-id job-run-id
```

Lorsque la destination de votre journal est définie surs`3://amzn-s3-demo-bucket/emr-serverless-hive/logs`, vous pouvez trouver les journaux de cette tâche spécifique exécutée souss`3://amzn-s3-demo-bucket/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id`.

Pour les applications Hive, EMR Serverless télécharge en permanence le pilote Hive dans le dossier et `HIVE_DRIVER` les journaux de tâches Tez dans `TEZ_TASK` le dossier de votre destination de journal S3. Une fois que le job a atteint son `SUCCEEDED` état, le résultat de votre requête Hive est disponible à l'emplacement Amazon S3 que vous avez spécifié dans le `monitoringConfiguration` champ de`configurationOverrides`.

Étape 4 : nettoyer

Lorsque vous aurez terminé de travailler sur ce didacticiel, pensez à supprimer les ressources que vous avez créées. Nous vous recommandons de libérer les ressources que vous n'avez pas l'intention de réutiliser.

Supprimer votre candidature

Pour supprimer une application, utilisez la commande suivante.

```
aws emr-serverless delete-application \
    --application-id application-id
```

Supprimer votre bucket de log S3

Pour supprimer votre compartiment de journalisation et de sortie S3, utilisez la commande suivante. Remplacez *amzn-s3-demo-bucket* par le nom réel du compartiment S3 créé dansPréparer le stockage pour EMR Serverless..

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive
aws s3api delete-bucket --bucket amzn-s3-demo-bucket
```

Supprimer votre rôle d'exécution des tâches

Pour supprimer le rôle d'exécution, détachez la politique du rôle. Vous pouvez ensuite supprimer à la fois le rôle et la politique.

```
aws iam detach-role-policy \
    --role-name EMRServerlessS3RuntimeRole \
    --policy-arn policy-arn
```

Pour supprimer le rôle, utilisez la commande suivante.

```
aws iam delete-role \
    --role-name EMRServerlessS3RuntimeRole
```

Pour supprimer la politique attachée au rôle, utilisez la commande suivante.

```
aws iam delete-policy \
    --policy-arn policy-arn
```

Pour d'autres exemples d'exécution de tâches Spark et Hive, consultez Utilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur etUtilisation de configurations Hive lorsque vous exécutez des tâches EMR sans serveur.

Interagissez avec une application EMR sans serveur et configurez-la

Cette section explique comment interagir avec votre application Amazon EMR Serverless avec le. AWS CLI Il décrit également la configuration d'une application, les personnalisations et les paramètres par défaut des moteurs Spark et Hive.

Rubriques

États de l'application

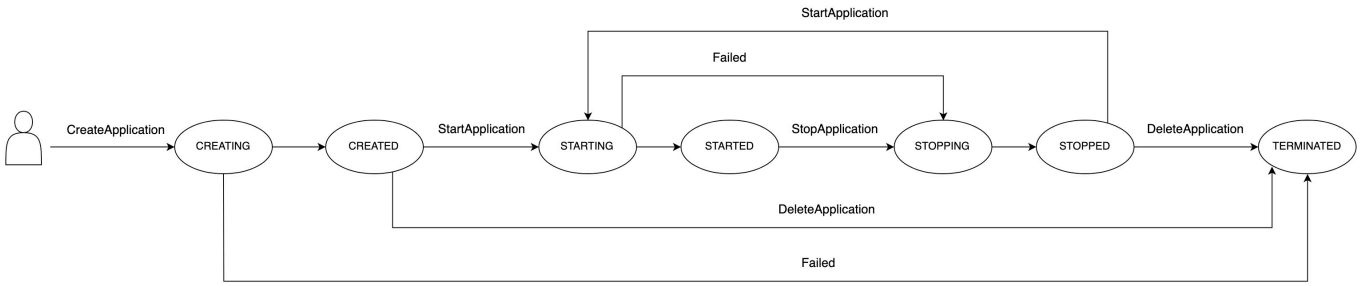
Lorsque vous créez une application avec EMR Serverless, l'application exécutée entre dans cet état. `CREATING` Elle passe ensuite par les états suivants jusqu'à ce qu'elle réussisse (se termine avec le code `0`) ou qu'elle échoue (se termine avec un code non nul).

Les applications peuvent avoir les états suivants :

State	Description
Création	L'application est en cours de préparation et n'est pas encore prête à être utilisée.
Créé	L'application a été créée mais n'a pas encore provisionné de capacité. Vous pouvez modifier

State	Description
	l'application pour modifier sa configuration initiale de capacité.
Démarrage en cours	L'application démarre et fournit de la capacité.
Commencé	L'application est prête à accepter de nouveaux emplois. L'application n'accepte les offres d'emploi que lorsqu'elle est dans cet état.
Arrêt en cours	Tous les travaux sont terminés et l'application est en train de libérer ses capacités.
Arrêté(e)	L'application est arrêtée et aucune ressource n'est en cours d'exécution sur l'application. Vous pouvez modifier l'application pour modifier sa configuration initiale de capacité.
Terminé	L'application a été résiliée et ne figure pas dans votre liste de candidatures.

Le schéma suivant illustre la trajectoire des états des applications EMR Serverless.



Création d'une application EMR sans serveur à partir de la console EMR Studio

À partir de la console EMR Studio, créez, accédez et gérez des applications EMR Serverless. Pour accéder à la console EMR Studio, suivez les instructions de la section Démarrage depuis la console.

Création d’une application

Sur la page Créer une application, créez une application EMR sans serveur en suivant ces étapes.

1. Dans le champ Nom, saisissez le nom que vous souhaitez donner à votre application.
2. Dans le champ Type, choisissez Spark ou Hive comme type d'application.
3. Dans le champ Version de version, choisissez le numéro de version EMR.
4. Dans les options d'architecture, choisissez l'architecture du jeu d'instructions à utiliser. Pour plus d’informations, consultez Options d'architecture sans serveur Amazon EMR.
   - arm64 — architecture ARM 64 bits ; pour utiliser les processeurs Graviton
   - x86_64 : architecture x86 64 bits ; pour utiliser des processeurs x86
5. Il existe deux options de configuration de l'application pour les champs restants : les paramètres par défaut et les paramètres personnalisés. Ces champs sont facultatifs.

   Paramètres par défaut — Les paramètres par défaut vous permettent de créer rapidement une application avec une capacité pré-initialisée. Cela inclut un pilote et un exécuteur pour Spark, et un pilote et un Tez Task pour Hive. Les paramètres par défaut n'activent pas la connectivité réseau à votre VPCs. L'application est configurée pour s'arrêter en cas d'inactivité pendant 15 minutes et démarre automatiquement lors de la soumission de la tâche.

   Paramètres personnalisés : les paramètres personnalisés vous permettent de modifier les propriétés suivantes.

   - Capacité pré-initialisée : nombre de pilotes et d'exécuteurs ou de travailleurs Hive Tez Task, et taille de chaque travailleur.
   - Limites d'application : capacité maximale d'une application.
   - Comportement de l'application : comportement de démarrage et d'arrêt automatique de l'application.
   - Connexions réseau : connectivité réseau aux ressources VPC.
   - Balises : balises personnalisées attribuées à l'application.

   Pour plus d'informations sur la capacité préinitialisée, les limites des applications et le comportement des applications, reportez-vous à. Configuration d'une application lorsque vous travaillez avec EMR Serverless Pour plus d'informations sur la connectivité réseau, reportez-

vous àConfiguration de l'accès VPC pour que les applications EMR sans serveur se connectent aux données.

6. Pour créer l'application, choisissez Create application.

Répertorier les applications depuis la console EMR Studio

Vous pouvez accéder à toutes les applications EMR Serverless existantes depuis la page Lister les applications. Vous pouvez choisir le nom d'une application pour accéder à la page de détails de cette application.

Gérez les applications depuis la console EMR Studio

Vous pouvez effectuer les actions suivantes sur une application depuis la page Répertorier les applications ou depuis la page Détails d'une application spécifique.

Démarrer l'application

Choisissez cette option pour démarrer manuellement une application.

Arrêter l'application

Choisissez cette option pour arrêter manuellement une application. Aucune tâche en cours d'exécution ne doit être arrêtée pour qu'une application puisse être arrêtée. Pour en savoir plus sur les transitions d'état des applications, reportez-vous àÉtats de l'application.

Configuration de l'application

Modifiez les paramètres facultatifs d'une application depuis la page Configurer l'application. Vous pouvez modifier la plupart des paramètres de l'application. Par exemple, modifiez l'étiquette de version d'une application pour la mettre à niveau vers une autre version d'Amazon EMR, ou passez de l'architecture x86_64 à arm64. Les autres paramètres facultatifs sont les mêmes que ceux de la section Paramètres personnalisés de la page Créer une application. Pour plus d'informations sur les paramètres de l'application, reportez-vous àCréation d’une application.

Supprimer l'application

Choisissez cette option pour supprimer manuellement une application. Vous devez arrêter une application pour la supprimer. Pour en savoir plus sur les transitions d'état des applications, reportez-vous àÉtats de l'application.

Interaction avec votre application EMR Serverless sur AWS CLI

À partir de AWS CLI, créez, décrivez et supprimez des applications individuelles. Vous pouvez également répertorier toutes vos applications afin d'y accéder en un coup d'œil. Cette section décrit comment effectuer ces actions. Pour d'autres opérations d'application, telles que le démarrage, l'arrêt et les mises à jour des applications, reportez-vous au manuel EMR Serverless API Reference. Pour des exemples d'utilisation de l'API EMR Serverless à l'aide de AWS SDK pour Java, reportez-vous aux exemples Java de notre référentiel. GitHub Pour des exemples d'utilisation de l'API EMR Serverless à l'aide de AWS SDK pour Python (Boto), reportez-vous aux exemples de Python dans notre référentiel. GitHub

Pour créer une application, utilisez`create-application`. Vous devez spécifier `SPARK` ou `HIVE` en tant qu'application`type`. Cette commande renvoie l'ARN, le nom et l'ID de l'application.

```
aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
--release-label release-version
```

Pour décrire une application, utilisez `get-application` et fournissez son`application-id`. Cette commande renvoie les configurations relatives à l'état et à la capacité de votre application.

```
aws emr-serverless get-application \
--application-id application-id
```

Pour répertorier toutes vos applications, appelez`list-applications`. Cette commande renvoie les mêmes propriétés que toutes vos applications`get-application`, mais inclut toutes les applications.

```
aws emr-serverless list-applications
```

Pour supprimer votre application, appelez `delete-application` et fournissez votre`application-id`.

```
aws emr-serverless delete-application \
--application-id application-id
```

Configuration d'une application lorsque vous travaillez avec EMR Serverless

Avec EMR Serverless, configurez les applications que vous utilisez. Par exemple, définissez la capacité maximale qu'une application peut atteindre, configurez la capacité préinitialisée pour que les conducteurs et les employés soient prêts à réagir, et spécifiez un ensemble commun de configurations d'exécution et de surveillance au niveau de l'application. Les pages suivantes décrivent comment configurer des applications lorsque vous utilisez EMR Serverless.

Rubriques

Comprendre le comportement des applications dans EMR Serverless

Cette section décrit le comportement de soumission des tâches, la configuration des capacités pour le dimensionnement et les paramètres de configuration des travailleurs pour EMR Serverless.

Comportement par défaut des applications

Démarrage automatique : une application est configurée par défaut pour démarrer automatiquement lors de la soumission de la tâche. Vous pouvez désactiver cette fonctionnalité.

Arrêt automatique : une application est configurée par défaut pour s'arrêter automatiquement lorsqu'elle est inactive pendant 15 minutes. Lorsqu'une application passe à `STOPPED` cet état, elle libère toute capacité préinitialisée configurée. Vous pouvez modifier le temps d'inactivité avant l'arrêt automatique d'une application ou désactiver cette fonctionnalité.

Maximum capacity (Capacité maximum)

Vous pouvez configurer la capacité maximale qu'une application peut atteindre. Vous pouvez spécifier votre capacité maximale en termes de processeur, de mémoire (Go) et de disque (Go).

> **Note**
>
> Il est recommandé de configurer votre capacité maximale de manière à ce qu'elle soit proportionnelle à la taille des travailleurs pris en charge en multipliant le nombre de

travailleurs par leur taille. Par exemple, si vous souhaitez limiter votre application à 50 serveurs dotés de 2 VCPUs, 16 Go de mémoire et 20 Go de disque, définissez votre capacité maximale à 100 VCPUs, 800 Go de mémoire et 1 000 Go de disque.

Configurations de travail supportées

Le tableau suivant répertorie les configurations de travail prises en charge et les tailles qui peuvent être spécifiées pour EMR Serverless. Configurez différentes tailles pour les pilotes et les exécuteurs en fonction des besoins de votre charge de travail.

Configurations et tailles des travailleurs

CPU	Mémoire	Stockage éphémère par défaut
1 vCPU	Minimum 2 Go, maximum 8 Go, par incréments de 1 Go	20 GO - 200 GO
2 vCPU	Minimum 4 Go, maximum 16 Go, par incréments de 1 Go	20 GO - 200 GO
4 vCPU	Minimum 8 Go, maximum 30 Go, par incréments de 1 Go	20 GO - 200 GO
8 vCPU	16 Go au minimum, 60 Go au maximum, par incréments de 4 Go	20 GO - 200 GO
16 vCPU	Minimum 32 Go, maximum 120 Go, par incréments de 8 Go	20 GO - 200 GO

Processeur — Chaque travailleur peut avoir 1, 2, 4, 8 ou 16 CPUs v.

Mémoire — Chaque travailleur dispose de mémoire, spécifiée en Go, dans les limites indiquées dans le tableau précédent. Les tâches Spark ont une surcharge de mémoire, ce qui signifie que la mémoire qu'elles utilisent est supérieure aux tailles de conteneur spécifiées. Cette surcharge est spécifiée avec les propriétés `spark.driver.memoryOverhead` et`spark.executor.memoryOverhead`.

La surcharge a une valeur par défaut de 10 % de la mémoire du conteneur, avec un minimum de 384 Mo. Vous devez tenir compte de ces frais généraux lorsque vous choisissez la taille des travailleurs.

Par exemple, si vous choisissez 4 V CPUs pour votre instance de travail et une capacité de stockage préinitialisée de 30 Go, définissez une valeur d'environ 27 Go comme mémoire d'exécuteur pour votre tâche Spark. Cela maximise l'utilisation de votre capacité pré-initialisée. La mémoire utilisable est de 27 Go, plus 10 % de 27 Go (2,7 Go), pour un total de 29,7 Go.

Disque : vous pouvez configurer chaque travailleur avec des disques de stockage temporaires d'une taille minimale de 20 Go et maximale de 200 Go. Vous ne payez que pour le stockage supplémentaire au-delà de 20 Go que vous configurez par utilisateur.

Capacité pré-initialisée pour travailler avec une application dans EMR Serverless

EMR Serverless fournit une fonctionnalité optionnelle qui permet au conducteur et aux employés de rester préinitialisés et prêts à réagir en quelques secondes. Cela crée efficacement un bassin de travailleurs chaleureux pour une application. Cette fonctionnalité est appelée capacité pré-initialisée. Pour configurer cette fonctionnalité, définissez le `initialCapacity` paramètre d'une application sur le nombre de travailleurs que vous souhaitez pré-initialiser. Avec une capacité de travail préinitialisée, les tâches démarrent immédiatement. C'est idéal lorsque vous souhaitez implémenter des applications itératives et des tâches urgentes.

La capacité préinitialisée permet de maintenir un pool de travailleurs prêts à exécuter des tâches et de démarrer des sessions en quelques secondes. Vous devrez payer pour des travailleurs préinitialisés provisionnés même lorsque l'application est inactive. Nous vous suggérons donc de l'activer pour les cas d'utilisation bénéficiant d'un temps de démarrage rapide et de la dimensionner pour une utilisation optimale des ressources. Les applications EMR Serverless s'arrêtent automatiquement lorsqu'elles sont inactives. Nous vous conseillons de conserver cette fonctionnalité activée lorsque vous utilisez des outils de travail préinitialisés afin d'éviter des frais imprévus.

Lorsque vous soumettez une tâche, si des collaborateurs `initialCapacity` sont disponibles, la tâche utilise ces ressources pour démarrer son exécution. Si ces travailleurs sont déjà utilisés par d'autres tâches, ou si la tâche nécessite plus de ressources que celles disponibles`initialCapacity`, l'application demande et obtient des travailleurs supplémentaires, dans la limite maximale des ressources définies pour l'application. Lorsqu'une tâche termine son exécution, elle libère les travailleurs qu'elle a utilisés et le nombre de ressources disponibles pour l'application revient à zéro`initialCapacity`. Une application conserve les ressources même après `initialCapacity` la fin de l'exécution des tâches. L'application libère des ressources

excédentaires au-delà du `initialCapacity` moment où les tâches n'en ont plus besoin pour s'exécuter.

La capacité pré-initialisée est disponible et prête à être utilisée au démarrage de l'application. La capacité pré-initialisée devient inactive lorsque l'application est arrêtée. Une application ne passe à l'`STARTED`état que si la capacité pré-initialisée demandée a été créée et est prête à être utilisée. Pendant toute la durée de l'`STARTED`état de l'application, EMR Serverless conserve la capacité préinitialisée disponible pour être utilisée ou utilisée par des tâches ou des charges de travail interactives. Cette fonctionnalité rétablit la capacité des conteneurs libérés ou défaillants. Cela permet de conserver le nombre de travailleurs spécifié par le `InitialCapacity` paramètre. L'état d'une application sans capacité préinitialisée peut immédiatement passer de à`CREATED`. `STARTED`

Vous pouvez configurer l'application pour libérer de la capacité préinitialisée si elle n'est pas utilisée pendant un certain temps, avec une durée par défaut de 15 minutes. Une candidature bloquée démarre automatiquement lorsque vous soumettez une nouvelle offre d'emploi. Vous pouvez définir ces configurations de démarrage et d'arrêt automatiques lorsque vous créez l'application, ou les modifier lorsque l'application est à `STOPPED` l'état `CREATED` ou.

Vous pouvez modifier les `InitialCapacity` nombres et spécifier des configurations de calcul telles que le processeur, la mémoire et le disque pour chaque travailleur. Comme vous ne pouvez pas apporter de modifications partielles, spécifiez toutes les configurations de calcul lorsque vous modifiez des valeurs. Vous ne pouvez modifier les configurations que lorsque l'application est à l'`STOPPED`état `CREATED` ou.

> **Note**
>
> Pour optimiser l'utilisation des ressources par votre application, nous vous suggérons d'aligner la taille de vos conteneurs sur les tailles de capacité de travail préinitialisées. Par exemple, si vous configurez la taille de votre exécuteur Spark à 2 CPUs et votre mémoire à 8 Go, mais que votre capacité de travail préinitialisée est de 4 CPUs avec 16 Go de mémoire, les exécuteurs Spark n'utilisent que la moitié des ressources des travailleurs lorsqu'ils sont affectés à cette tâche.

Personnalisation de la capacité pré-initialisée pour Spark et Hive

Vous pouvez personnaliser davantage la capacité préinitialisée pour les charges de travail exécutées sur des frameworks de mégadonnées spécifiques. Par exemple, lorsqu'une charge de travail s'exécute sur Apache Spark, spécifiez combien de travailleurs démarrent en tant que pilotes et

combien démarrent en tant qu'exécuteurs. De même, lorsque vous utilisez Apache Hive, spécifiez le nombre de travailleurs qui démarrent en tant que pilotes Hive et combien doivent exécuter les tâches Tez.

Configuration d'une application exécutant Apache Hive avec une capacité pré-initialisée

La demande d'API suivante crée une application exécutant Apache Hive sur la base de la version emr-6.6.0 d'Amazon EMR. L'application démarre avec 5 pilotes Hive préinitialisés, chacun avec 2 vCPU et 4 Go de mémoire, et 50 programmes de travail Tez pré-initialisés, chacun avec 4 vCPU et 8 Go de mémoire. Lorsque les requêtes Hive s'exécutent sur cette application, elles utilisent d'abord les travailleurs pré-initialisés et commencent à s'exécuter immédiatement. Si tous les travailleurs pré-initialisés sont occupés et que d'autres tâches Hive sont soumises, l'application peut atteindre un total de 400 vCPU et 1 024 Go de mémoire. Vous pouvez éventuellement omettre la capacité du travailleur `DRIVER` ou du `TEZ_TASK` travailleur.

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "TEZ_TASK": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

Configuration d'une application exécutant Apache Spark avec une capacité pré-initialisée

La demande d'API suivante crée une application qui exécute Apache Spark 3.2.0 sur la base de la version 6.6.0 d'Amazon EMR. L'application démarre avec 5 pilotes Spark préinitialisés, chacun avec 2 vCPU et 4 Go de mémoire, et 50 exécuteurs pré-initialisés, chacun avec 4 vCPU et 8 Go de mémoire. Lorsque les tâches Spark s'exécutent sur cette application, elles utilisent d'abord les travailleurs pré-initialisés et commencent à s'exécuter immédiatement. Si tous les travailleurs préinitialisés sont occupés et que d'autres tâches Spark sont soumises, l'application peut atteindre un total de 400 vCPU et 1 024 Go de mémoire. Vous pouvez éventuellement omettre la capacité pour le `DRIVER` ou le`EXECUTOR`.

> **Note**
>
> Spark ajoute une surcharge de mémoire configurable, avec une valeur par défaut de 10 %, à la mémoire demandée pour le pilote et les exécuteurs. Pour que les tâches utilisent des travailleurs préinitialisés, la configuration initiale de la capacité de mémoire doit être supérieure à la mémoire demandée par la tâche et la surcharge.

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
```

```
}'
```

Configuration d'application par défaut pour EMR Serverless

Vous pouvez définir un ensemble commun de configurations d'exécution et de surveillance au niveau de l'application pour toutes les tâches que vous soumettez dans le cadre de la même application. Cela réduit la surcharge supplémentaire associée à la nécessité de soumettre les mêmes configurations pour chaque tâche.

Vous pouvez modifier les configurations aux moments suivants :

- Déclarez les configurations au niveau de l'application lors de la soumission de la tâche.
- Remplacez les configurations par défaut lors de l'exécution du job.

Les sections suivantes fournissent plus de détails et fournissent un exemple pour un contexte plus détaillé.

Déclarer des configurations au niveau de l'application

Vous pouvez spécifier les propriétés de journalisation et de configuration d'exécution au niveau de l'application pour les tâches que vous soumettez dans le cadre de l'application.

`monitoringConfiguration`

Pour spécifier les configurations du journal pour les tâches que vous soumettez avec l'application, utilisez le `monitoringConfiguration`champ. Pour plus d'informations sur la journalisation pour EMR Serverless, reportez-vous à. Stockage des journaux

`runtimeConfiguration`

Pour spécifier les propriétés de configuration d'exécution, par exemple`spark-defaults`, fournissez un objet de configuration dans le `runtimeConfiguration` champ. Cela affecte les configurations par défaut de toutes les tâches que vous soumettez avec l'application. Pour plus d'informations, consultez Paramètre de remplacement de la configuration Hive et Paramètre de remplacement de la configuration Spark.

Les classifications de configuration disponibles varient en fonction de la version EMR Serverless spécifique. Par exemple, les classifications pour Log4j personnalisées ne `spark-executor-log4j2` sont disponibles qu'avec les versions 6.8.0 `spark-driver-log4j2` et supérieures.

Pour obtenir la liste des propriétés spécifiques à l'application, reportez-vous à Propriétés de la tâche Spark et. Propriétés des tâches Hive

Vous pouvez également configurer les propriétés Apache Log4j2, AWS Secrets Manager pour la protection des données, et le runtime Java 17 au niveau de l'application.

Pour transmettre les secrets de Secrets Manager au niveau de l'application, associez la politique suivante aux utilisateurs et aux rôles qui doivent créer ou mettre à jour des applications EMR Serverless avec des secrets.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "SecretsManagerPolicy",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret-
name-123abc"
      ]
    },
    {
      "Sid": "KMSDecryptPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      ]
    }
  ]
}
```

Pour plus d'informations sur la création de politiques personnalisées pour les secrets, reportez-vous aux exemples de politiques d' AWS Secrets Manager autorisations du Guide de AWS Secrets Manager l'utilisateur.

> **Note**
>
> `runtimeConfiguration`Ce que vous spécifiez au niveau de l'application `applicationConfiguration` correspond à l'`StartJobRun`API.

Exemple de déclaration

L'exemple suivant montre comment déclarer des configurations par défaut avec`create-application`.

```
aws emr-serverless create-application \
    --release-label release-version  \
    --type SPARK \
    --name my-application-name \
    --runtime-configuration '[
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.driver.cores": "4",
                "spark.executor.cores": "2",
                "spark.driver.memory": "8G",
                "spark.executor.memory": "8G",
                "spark.executor.instances": "2",

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver",
                "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name",
                "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-
name",
                "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretID"
            }
        },
        {
            "classification": "spark-driver-log4j2",
            "properties": {
```

```
                "rootLogger.level":"error",
                "logger.IdentifierForClass.name": "classpathForSettingLogger",
                "logger.IdentifierForClass.level": "info"
            }
        }
    ]' \
    --monitoring-configuration '{
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

Remplacer les configurations lors de l'exécution d'une tâche

Vous pouvez définir des remplacements de configuration pour la configuration de l'application et la configuration de surveillance à l'aide de l'`StartJobRun`API. EMR Serverless fusionne ensuite les configurations que vous spécifiez au niveau de l'application et au niveau de la tâche afin de déterminer les configurations pour l'exécution de la tâche.

Le niveau de granularité lors de la fusion est le suivant :

- **`ApplicationConfiguration`**- Type de classification, par exemple`spark-defaults`.
- **`MonitoringConfiguration`**- Type de configuration, par exemple`s3MonitoringConfiguration`.

> **Note**
>
> La priorité des configurations que vous fournissez `StartJobRun`remplace les configurations que vous fournissez au niveau de l'application.

Pour plus d'informations sur les classements prioritaires, reportez-vous à Paramètre de remplacement de la configuration Hive etParamètre de remplacement de la configuration Spark.

Lorsque vous démarrez une tâche, si vous ne spécifiez pas de configuration particulière, elle est héritée de l'application. Si vous déclarez les configurations au niveau de la tâche, vous pouvez effectuer les opérations suivantes :

- Remplacer une configuration existante : fournissez le même paramètre de configuration dans la `StartJobRun` demande avec vos valeurs de remplacement.
- Ajouter une configuration supplémentaire : ajoutez le nouveau paramètre de configuration dans la `StartJobRun` demande avec les valeurs que vous souhaitez spécifier.
- Supprimer une configuration existante : pour supprimer une configuration d'exécution d'application, fournissez la clé de la configuration que vous souhaitez supprimer et transmettez une déclaration vide `{}` pour la configuration. Nous ne recommandons pas de supprimer les classifications contenant les paramètres nécessaires à l'exécution d'une tâche. Par exemple, si vous essayez de supprimer les propriétés requises pour une tâche Hive, la tâche échouera.

  Pour supprimer une configuration de surveillance des applications, utilisez la méthode appropriée pour le type de configuration concerné :

  - **`cloudWatchLoggingConfiguration`**- Pour le supprimer`cloudWatchLogging`, transmettez le drapeau activé en tant que`false`.
  - **`managedPersistenceMonitoringConfiguration`**- Pour supprimer les paramètres de persistance gérés et revenir à l'état activé par défaut, transmettez une déclaration vide `{}` pour la configuration.
  - **`s3MonitoringConfiguration`**- Pour supprimer`s3MonitoringConfiguration`, transmettez une déclaration vide `{}` pour la configuration.

Exemple de dérogation

L'exemple suivant montre les différentes opérations que vous pouvez effectuer lors de la soumission d'une tâche sur`start-job-run`.

```
aws emr-serverless start-job-run \
    --application-id your-application-id \
    --execution-role-arn your-job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket1/
wordcount_output"]
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [
```

```
            {
                // Override existing configuration for spark-defaults in the
application
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.cores": "2",
                    "spark.executor.cores": "1",
                    "spark.driver.memory": "4G",
                    "spark.executor.memory": "4G"
                }
            },
            {
                // Add configuration for spark-executor-log4j2
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level": "error",
                    "logger.IdentifierForClass.name": "classpathForSettingLogger",
                    "logger.IdentifierForClass.level": "info"
                }
            },
            {
                // Remove existing configuration for spark-driver-log4j2 from the
application
                "classification": "spark-driver-log4j2",
                "properties": {}
            }
        ],
        "monitoringConfiguration": {
            "managedPersistenceMonitoringConfiguration": {
                // Override existing configuration for managed persistence
                "enabled": true
            },
            "s3MonitoringConfiguration": {
                // Remove configuration of S3 monitoring
            },
            "cloudWatchLoggingConfiguration": {
                // Add configuration for CloudWatch logging
                "enabled": true
            }
        }
    }'
```

Au moment de l'exécution de la tâche, les classifications et configurations suivantes s'appliqueront en fonction du classement de priorité décrit dans Paramètre de remplacement de la configuration Hive etParamètre de remplacement de la configuration Spark.

- La classification `spark-defaults` sera mise à jour avec les propriétés spécifiées au niveau du poste. Seules les propriétés incluses sont prises en `StartJobRun` compte pour cette classification.
- La classification `spark-executor-log4j2` sera ajoutée à la liste de classifications existante.
- La classification `spark-driver-log4j2` sera supprimée.
- Les configurations pour `managedPersistenceMonitoringConfiguration` seront mises à jour avec les configurations au niveau du travail.
- Les configurations pour `s3MonitoringConfiguration` seront supprimées.
- Les configurations pour `cloudWatchLoggingConfiguration` seront ajoutées aux configurations de surveillance existantes.

Personnalisation d'une image EMR sans serveur

À partir d'Amazon EMR 6.9.0, utilisez des images personnalisées pour regrouper les dépendances des applications et les environnements d'exécution dans un conteneur unique avec Amazon EMR Serverless. Cela simplifie la gestion des dépendances de charge de travail et rend vos packages plus portables. Lorsque vous personnalisez votre image EMR Serverless, elle offre les avantages suivants :

- Installe et configure des packages optimisés pour vos charges de travail. Ces packages ne sont pas largement disponibles dans la distribution publique des environnements d'exécution Amazon EMR.
- Intègre EMR Serverless aux processus de création, de test et de déploiement existants au sein de votre organisation, y compris le développement et les tests locaux.
- Applique 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.
- Vous permet d'utiliser vos propres versions de JDK et Python pour vos applications.

EMR Serverless fournit des images qui vous serviront de base lorsque vous créez vos propres images. L'image de base fournit les fichiers JAR, la configuration et les bibliothèques essentiels pour que l'image interagisse avec EMR Serverless. Vous pouvez trouver l'image de base dans la galerie

publique Amazon ECR. Utilisez l'image correspondant au type d'application (Spark ou Hive) et à la version de votre version. Par exemple, si vous créez une application sur Amazon EMR version 6.9.0, utilisez les images suivantes.

Type	Image
Spark	`public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest`
Hive	`public.ecr.aws/emr-serverless/hive/emr-6.9.0:latest`

Conditions préalables

Avant de créer une image personnalisée EMR Serverless, remplissez ces conditions préalables.

1. Créez un référentiel Amazon ECR dans le même référentiel Région AWS que celui que vous utilisez pour lancer des applications EMR Serverless. Pour créer un référentiel privé Amazon ECR, reportez-vous à la section Création d'un référentiel privé.
2. Pour autoriser les utilisateurs à accéder à votre référentiel Amazon ECR, ajoutez les politiques suivantes aux utilisateurs et aux rôles qui créent ou mettent à jour des applications EMR sans serveur avec des images provenant de ce référentiel.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ECRRepositoryListGetPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:DescribeImages"
      ],
      "Resource": [
        "arn:aws:ecr:*:123456789012:repository/my-repo"
      ]
    }
```

```
  ]
}
```

Pour plus d'exemples de politiques basées sur l'identité Amazon ECR, consultez les exemples de politiques basées sur l'identité Amazon Elastic Container Registry.

Étape 1 : Création d'une image personnalisée à partir d'images de base EMR Serverless

Créez d'abord un Dockerfile qui commence par une `FROM` instruction utilisant votre image de base préférée. Après les `FROM` instructions, incluez les modifications que vous souhaitez apporter à l'image. L'image de base définit automatiquement le `USER` à`hadoop`. Ce paramètre ne dispose pas d'autorisations pour toutes les modifications que vous incluez. Pour contourner le problème, réglez le `USER` paramètre sur`root`, modifiez votre image, puis `USER` redéfinissez le paramètre sur. `hadoop:hadoop` Pour consulter des exemples de cas d'utilisation courants, reportez-vous àUtilisation d'images personnalisées avec EMR Serverless.

```
# Dockerfile
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root
# MODIFICATIONS GO HERE

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

Une fois que vous avez le Dockerfile, créez l'image à l'aide de la commande suivante.

```
# build the docker image
docker build . -t aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository[:tag]or[@digest]
```

Étape 2 : Valider l'image localement

EMR Serverless fournit un outil hors ligne qui permet de vérifier statiquement votre image personnalisée afin de valider les fichiers de base, les variables d'environnement et les configurations d'image correctes. Pour plus d'informations sur l'installation et l'exécution de l'outil, reportez-vous à la CLI Amazon EMR Serverless Image. GitHub

Après avoir installé l'outil, exécutez la commande suivante pour valider une image :

```
amazon-emr-serverless-image \
validate-image -r emr-6.9.0 -t spark \
-i aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

Le résultat est similaire à ce qui suit.

```
Amazon EMR Serverless - Image CLI
Version: 0.0.1
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: 9e2f4359cf5beb466a8a2ed047ab61c9d37786c555655fc122272758f761b41a
[INFO] Created On: 2022-12-02T07:46:42.586249984Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to  : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] HADOOP_HOME is set with value: /usr/lib/hadoop : PASS
[INFO] HADOOP_LIBEXEC_DIR is set with value: /usr/lib/hadoop/libexec : PASS
[INFO] HADOOP_USER_HOME is set with value: /home/hadoop : PASS
[INFO] HADOOP_YARN_HOME is set with value: /usr/lib/hadoop-yarn : PASS
[INFO] HIVE_HOME is set with value: /usr/lib/hive : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] TEZ_HOME is set with value: /usr/lib/tez : PASS
[INFO] YARN_HOME is set with value: /usr/lib/hadoop-yarn : 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 hadoop-yarn-jars in /usr/lib/hadoop-yarn: PASS
[INFO] File Structure Test for hive-bin-files in /usr/bin: PASS
[INFO] File Structure Test for hive-jars in /usr/lib/hive/lib: PASS
[INFO] File Structure Test for java-bin in /etc/alternatives/jre/bin: PASS
[INFO] File Structure Test for tez-jars in /usr/lib/tez: PASS
----------------------------------------------------------------
Overall Custom Image Validation Succeeded.
----------------------------------------------------------------
```

Étape 3 : Chargez l'image dans votre référentiel Amazon ECR

Transférez votre image Amazon ECR vers votre référentiel Amazon ECR à l'aide des commandes suivantes. Assurez-vous de disposer des autorisations IAM appropriées pour transférer l'image vers votre référentiel. Pour plus d'informations, consultez la section Envoyer une image dans le guide de l'utilisateur Amazon ECR.

```
# login to ECR repo
aws ecr get-login-password --region region | docker login --username AWS --password-
stdin aws-account-id.dkr.ecr.region.amazonaws.com

# push the docker image
docker push aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

Étape 4 : créer ou mettre à jour une application avec des images personnalisées

Choisissez l' AWS Management Console onglet ou AWS CLI l'onglet en fonction de la façon dont vous souhaitez lancer votre application, puis effectuez les étapes suivantes.

Console

1. Connectez-vous à la console EMR Studio à l'adresse /emr. https://console.aws.amazon.com Accédez à votre application ou créez-en une nouvelle en suivant les instructions de la section Créer une application.
2. Pour spécifier des images personnalisées lorsque vous créez ou mettez à jour une application EMR Serverless, sélectionnez Paramètres personnalisés dans les options de configuration de l'application.
3. Dans la section Paramètres d'image personnalisés, cochez la case Utiliser l'image personnalisée avec cette application.
4. Collez l'URI de l'image Amazon ECR dans le champ Image URI. EMR Serverless utilise cette image pour tous les types de travailleurs de l'application. Vous pouvez également choisir Différentes images personnalisées et coller une image Amazon ECR différente URIs pour chaque type de travailleur.

CLI

- Créez une application avec le `image-configuration` paramètre. EMR Serverless applique ce paramètre à tous les types de travailleurs.

```
aws emr-serverless create-application \
--release-label emr-6.9.0 \
--type SPARK \
--image-configuration '{
```

```
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'
```

Pour créer une application avec des paramètres d'image différents pour chaque type de travailleur, utilisez le `worker-type-specifications` paramètre.

```
aws emr-serverless create-application \
--release-label emr-6.9.0 \
--type SPARK \
--worker-type-specifications '{
    "Driver": {
        "imageConfiguration": {
            "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
        }
    },
    "Executor" : {
        "imageConfiguration": {
            "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
        }
    }
}'
```

Pour mettre à jour une application, utilisez le `image-configuration` paramètre. EMR Serverless applique ce paramètre à tous les types de travailleurs.

```
aws emr-serverless update-application \
--application-id application-id \
--image-configuration '{
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'
```

Étape 5 : Autoriser EMR Serverless à accéder au référentiel d'images personnalisé

Ajoutez la politique de ressources suivante au référentiel Amazon ECR pour permettre au principal du service EMR Serverless d'utiliser `get` les demandes`describe`, `download` et provenant de ce référentiel.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EmrServerlessCustomImageSupport",
      "Effect": "Allow",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": "arn:aws:ecr:*:123456789012:repository/my-repo",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
*"
        }
      }
    }
  ]
}
```

La meilleure pratique en matière de sécurité consiste à ajouter une clé de `aws:SourceArn` condition à la politique du référentiel. La clé de condition globale IAM `aws:SourceArn` garantit qu'EMR Serverless utilise le référentiel uniquement pour un ARN d'application. Pour plus d'informations sur les politiques relatives aux référentiels Amazon ECR, consultez la section Création d'un référentiel privé.

Considérations et restrictions

Lorsque vous travaillez avec des images personnalisées, tenez compte des points suivants :

- Utilisez l'image de base correcte qui correspond au type (Spark ou Hive) et à l'étiquette de version (par exemple`emr-6.9.0`) de votre application.
- EMR Serverless ignore les `[ENTRYPOINT]` instructions contenues dans le `[CMD]` fichier Docker. Utilisez les instructions courantes du fichier Docker, telles que `[COPY][RUN]`, et`[WORKDIR]`.
- Ne modifiez pas les variables `JAVA_HOME` `SPARK_HOME` d'`HIVE_HOME`environnement `TEZ_HOME` lorsque vous créez une image personnalisée.
- La taille des images personnalisées ne peut pas dépasser 10 Go.
- Si vous modifiez des fichiers binaires ou des fichiers JAR dans les images de base d'Amazon EMR, cela peut entraîner l'échec du lancement d'applications ou de tâches.
- Le référentiel Amazon ECR doit se trouver dans le même emplacement Région AWS que celui que vous utilisez pour lancer les applications EMR Serverless.

Configuration de l'accès VPC pour que les applications EMR sans serveur se connectent aux données

Vous pouvez configurer des applications EMR sans serveur pour qu'elles se connectent à vos magasins de données au sein de votre VPC, tels que les clusters Amazon Redshift, les bases de données Amazon RDS ou les compartiments Amazon S3 dotés de points de terminaison VPC. Votre application EMR Serverless dispose d'une connectivité sortante vers les magasins de données de votre VPC. Par défaut, EMR Serverless bloque à la fois l'accès entrant à vos applications et l'accès Internet sortant pour renforcer la sécurité.

> **Note**
>
> Vous devez configurer l'accès VPC si vous souhaitez utiliser une base de données de métastore Hive externe pour votre application. Pour plus d'informations sur la configuration d'un métastore Hive externe, reportez-vous à la section Configuration du métastore.

Création d’une application

Sur la page Créer une application, choisissez des paramètres personnalisés et spécifiez le VPC, les sous-réseaux et les groupes de sécurité que les applications EMR Serverless peuvent utiliser.

VPCs

Choisissez le nom du cloud privé virtuel (VPC) qui contient vos magasins de données. La page Créer une application répertorie toutes VPCs les applications que vous avez choisies Région AWS.

Subnets

Choisissez les sous-réseaux du VPC qui contient votre magasin de données. La page Créer une application répertorie tous les sous-réseaux des magasins de données de votre VPC. Les sous-réseaux publics et privés sont pris en charge. Vous pouvez transmettre des sous-réseaux privés ou publics à vos applications. Le choix d'avoir un sous-réseau public ou privé comporte quelques considérations connexes à prendre en compte.

Pour les sous-réseaux privés :

- Les tables de routage associées ne doivent pas comporter de passerelles Internet.
- Pour la connectivité sortante à Internet, si nécessaire, configurez les routes sortantes à l'aide d'une passerelle NAT. Pour configurer une passerelle NAT, reportez-vous à la section Passerelles NAT.
- Pour la connectivité Amazon S3, configurez une passerelle NAT ou un point de terminaison VPC. Pour configurer un point de terminaison VPC S3, reportez-vous à la section Créer un point de terminaison de passerelle.
- Si vous configurez un point de terminaison VPC S3 et que vous associez une politique de point de terminaison pour contrôler l'accès, suivez les instructions de la section Logging for EMR Serverless with managed storage afin d'autoriser EMR Serverless à stocker et à diffuser les journaux des applications.
- Pour la connectivité à d'autres Services AWS entités extérieures au VPC, comme Amazon DynamoDB, configurez des points de terminaison VPC ou une passerelle NAT. Pour configurer les points de terminaison VPC pour Services AWS, reportez-vous à la section Utilisation des points de terminaison VPC.

> **Note**
>
> Lorsque vous configurez une application Amazon EMR Serverless dans un sous-réseau privé, nous vous suggérons de configurer également des points de terminaison VPC pour Amazon S3. Si votre application EMR Serverless se trouve dans un sous-réseau privé sans points de terminaison VPC pour Amazon S3, vous devez payer des frais de passerelle NAT supplémentaires associés au trafic S3. Cela est dû au fait que le trafic entre votre application EMR et Amazon S3 ne restera pas dans votre VPC lorsque les points de terminaison du VPC ne sont pas configurés.

Pour les sous-réseaux publics :

- Ils ont une route vers un Internet Gateway.
- Vous devez vous assurer que les groupes de sécurité sont correctement configurés pour contrôler le trafic sortant.

Les collaborateurs peuvent se connecter aux magasins de données de votre VPC par le biais du trafic sortant. Par défaut, EMR Serverless bloque l'accès entrant aux travailleurs. Il s'agit d'améliorer la sécurité.

Lorsque vous l'utilisez AWS Config, EMR Serverless crée un enregistrement d'élément Elastic Network Interface pour chaque collaborateur. Pour éviter les coûts liés à cette ressource, pensez `AWS::EC2::NetworkInterface` à la désactiver AWS Config.

> **Note**
>
> Nous vous conseillons de sélectionner plusieurs sous-réseaux dans plusieurs zones de disponibilité. Cela est dû au fait que les sous-réseaux que vous choisissez déterminent les zones de disponibilité disponibles pour le lancement d'une application EMR sans serveur. Chaque travailleur consomme une adresse IP sur le sous-réseau où il est lancé. Assurez-vous que les sous-réseaux spécifiés possèdent suffisamment d'adresses IP pour le nombre de travailleurs que vous prévoyez de lancer. Pour plus d'informations sur la planification des sous-réseaux, reportez-vous àthe section called “Bonnes pratiques pour la planification des sous-réseaux”.

Considérations et limites relatives aux sous-réseaux

- L'EMR Serverless avec sous-réseaux publics ne prend pas en charge Lake Formation. AWS
- Le trafic entrant n'est pas pris en charge pour les sous-réseaux publics.

Groupes de sécurité

Choisissez un ou plusieurs groupes de sécurité qui peuvent communiquer avec vos banques de données. La page Créer une application répertorie tous les groupes de sécurité de votre VPC. EMR Serverless associe ces groupes de sécurité à des interfaces réseau élastiques associées à vos sous-réseaux VPC.

> **Note**
>
> Nous vous conseillons de créer un groupe de sécurité distinct pour les applications EMR Serverless. EMR Serverless ne vous permet pas de créer/mettre à jour/démarrer une application si les groupes de sécurité ont des ports ouverts à l'Internet public sur 0.0.0.0/0 ou sur la plage : :/0. Cela améliore la sécurité, l'isolation et rend la gestion des règles réseau plus efficace. Par exemple, cela bloque le trafic inattendu destiné aux employés possédant des adresses IP publiques. Pour communiquer avec les clusters Amazon Redshift, par exemple, définissez les règles de trafic entre les groupes de sécurité Redshift et EMR Serverless, comme illustré dans l'exemple de la section suivante.

Example Exemple — Communication avec les clusters Amazon Redshift

1. Ajoutez une règle pour le trafic entrant au groupe de sécurité Amazon Redshift depuis l'un des groupes de sécurité EMR Serverless.

Type	Protocole	Plage de ports	Source
Tous les TCP	TCP	5439	`emr-serverless-security-group`

2. Ajoutez une règle pour le trafic sortant depuis l'un des groupes de sécurité EMR Serverless. Procédez de deux manières différentes. Tout d'abord, ouvrez le trafic sortant à tous les ports.

Type	Protocole	Plage de ports	Destination
Tout le trafic	TCP	ALL	0.0.0.0/0

Vous pouvez également restreindre le trafic sortant vers les clusters Amazon Redshift. Cela n'est utile que lorsque l'application doit communiquer avec les clusters Amazon Redshift et rien d'autre.

Type	Protocole	Plage de ports	Source
Tous les TCP	TCP	5439	`redshift-security-group`

Configuration de l'application

Vous pouvez modifier la configuration réseau d'une application EMR Serverless existante à partir de la page Configurer l'application.

Accéder aux détails de l'exécution des tâches

Sur la page détaillée de l'exécution du job, accédez au sous-réseau utilisé par votre job pour une exécution spécifique. Notez qu'une tâche ne s'exécute que dans un sous-réseau sélectionné parmi les sous-réseaux spécifiés.

Bonnes pratiques pour la planification des sous-réseaux

AWS les ressources sont créées dans un sous-réseau qui est un sous-ensemble d'adresses IP disponibles dans un Amazon VPC. Par exemple, un VPC doté d'un masque réseau /16 possède jusqu'à 65 536 adresses IP disponibles qui peuvent être divisées en plusieurs réseaux plus petits à l'aide de masques de sous-réseau. Par exemple, vous pouvez diviser cette plage en deux sous-réseaux, chacun utilisant le masque /17 et 32 768 adresses IP disponibles. Un sous-réseau réside dans une zone de disponibilité et ne peut pas s'étendre sur plusieurs zones.

Les sous-réseaux doivent être conçus en tenant compte des limites de dimensionnement de votre application EMR Serverless. Par exemple, si votre application demande 4 processeurs virtuels et

peut évoluer jusqu'à 4 000 processeurs virtuels, votre application nécessite au maximum 1 000 travailleurs pour un total de 1 000 interfaces réseau. Nous vous conseillons de créer des sous-réseaux dans plusieurs zones de disponibilité. Cela permet à EMR Serverless de réessayer votre tâche ou de fournir une capacité préinitialisée dans une autre zone de disponibilité dans le cas peu probable d'une défaillance d'une zone de disponibilité. Par conséquent, chaque sous-réseau d'au moins deux zones de disponibilité doit avoir plus de 1 000 adresses IP disponibles.

Vous avez besoin de sous-réseaux dont la taille de masque est inférieure ou égale à 22 pour approvisionner 1 000 interfaces réseau. Tout masque supérieur à 22 ne répond pas à cette exigence. Par exemple, un masque de sous-réseau de /23 fournit 512 adresses IP, tandis qu'un masque de /22 fournit 1024 et un masque de /21 fournit 2048 adresses IP. Vous trouverez ci-dessous un exemple de 4 sous-réseaux avec un masque /22 dans un VPC de masque réseau /16 qui peuvent être alloués à différentes zones de disponibilité. Il existe une différence de cinq entre les adresses IP disponibles et utilisables, car les quatre premières adresses IP et la dernière adresse IP de chaque sous-réseau sont réservées par AWS.

ID de sous-réseau (subnet)	Adresse du sous-réseau	Masque de sous-réseau	Plage d'adresses IP	Adresses IP disponibles	Adresses IP utilisables
1	10,0.0.0	255,255,252,0/22	10.0.0.0 - 10.0.3.255	1,024	1 019
2	10.0.4.0	255,255,252,0/22	10.0.4.0 - 10.0.7.255	1,024	1 019
3	10,0.8.0	255,255,252,0/22	10,0.8.0 - 10,0.11.255	1,024	1 019
4	10,0.12,0	255,255,252,0/22	10,0.12,0 - 10,0.15,255	1,024	1 019

Vous devez évaluer si votre charge de travail convient le mieux à des travailleurs de plus grande taille. L'utilisation de travailleurs de plus grande taille nécessite moins d'interfaces réseau. Par exemple, l'utilisation de processeurs virtuels avec une limite de mise à l'échelle des applications de 4 000 processeurs virtuels nécessite au maximum 250 travailleurs pour un total de 250 adresses IP disponibles pour provisionner les interfaces réseau. Vous avez besoin de sous-réseaux répartis

dans plusieurs zones de disponibilité avec une taille de masque inférieure ou égale à 24 pour approvisionner 250 interfaces réseau. Toute taille de masque supérieure à 24 propose moins de 250 adresses IP.

Si vous partagez des sous-réseaux entre plusieurs applications, chaque sous-réseau doit être conçu en tenant compte des limites d'échelle collectives de toutes vos applications. Par exemple, si 3 applications demandent 4 processeurs virtuels et que chacune peut évoluer jusqu'à 4 000 processeurs virtuels avec un quota basé sur le service de 12 000 processeurs virtuels au niveau du compte, chaque sous-réseau nécessite 3 000 adresses IP disponibles. Si le VPC que vous souhaitez utiliser ne dispose pas d'un nombre suffisant d'adresses IP, essayez d'augmenter le nombre d'adresses IP disponibles. Vous pouvez effectuer cette opération en associant des blocs d'adresse CIDR (Routage inter-domaines sans classe) supplémentaires avec votre VPC. Pour plus d'informations, consultez la section Associer des blocs IPv4 CIDR supplémentaires à votre VPC dans le guide de l'utilisateur Amazon VPC.

Vous pouvez utiliser l'un des nombreux outils disponibles en ligne pour générer rapidement des définitions de sous-réseaux et passer en revue la plage d'adresses IP disponibles.

Options d'architecture sans serveur Amazon EMR

L'architecture du jeu d'instructions de votre application Amazon EMR Serverless détermine le type de processeurs que l'application utilise pour exécuter la tâche. Amazon EMR propose deux options d'architecture pour votre application : x86_64 et arm64. EMR Serverless est automatiquement mis à jour vers la dernière génération d'instances dès qu'elles sont disponibles, afin que vos applications puissent utiliser les nouvelles instances sans que vous ayez besoin d'efforts supplémentaires.

Rubriques

- Utilisation de l'architecture x86_64
- Utilisation de l'architecture arm64 (Graviton)
- Lancement de nouvelles applications avec le support de Graviton
- Configuration des applications existantes pour utiliser Graviton
- Considérations relatives à l'utilisation de Graviton

Utilisation de l'architecture x86_64

L'architecture x86_64 est également connue sous le nom de x86 64 bits ou x64. x86_64 est l'option par défaut pour les applications EMR sans serveur. Cette architecture utilise des processeurs x86 et est compatible avec la plupart des outils et bibliothèques tiers.

La plupart des applications sont compatibles avec la plate-forme matérielle x86 et peuvent fonctionner correctement sur l'architecture x86_64 par défaut. Toutefois, si votre application est compatible avec ARM 64 bits, passez à arm64 pour utiliser les processeurs Graviton afin d'améliorer les performances, la puissance de calcul et la mémoire. Exécuter des instances sur une architecture arm64 coûte moins cher que lorsque vous exécutez des instances de taille égale sur une architecture x86.

Utilisation de l'architecture arm64 (Graviton)

AWS Les processeurs Graviton sont conçus sur mesure AWS avec des cœurs ARM Neoverse 64 bits et tirent parti de l'architecture arm64 (également connue sous le nom d'Arch64 ou ARM 64 bits). La gamme de processeurs AWS Graviton disponible sur EMR Serverless inclut les processeurs Graviton3 et Graviton2. Ces processeurs offrent un rapport prix/performances supérieur pour les charges de travail Spark et Hive par rapport aux charges de travail équivalentes exécutées sur l'architecture x86_64. EMR Serverless utilise automatiquement la dernière génération de processeurs lorsqu'elle est disponible, sans aucun effort de votre part pour passer à la dernière génération de processeurs.

Lancement de nouvelles applications avec le support de Graviton

Utilisez l'une des méthodes suivantes pour lancer une application utilisant l'architecture arm64.

AWS CLI

Pour lancer une application à l'aide de processeurs Graviton depuis AWS CLI, `ARM64` spécifiez-le comme `architecture` paramètre dans l'`create-application`API. Indiquez les valeurs appropriées pour votre application dans les autres paramètres.

```
aws emr-serverless create-application \
 --name my-graviton-app \
 --release-label emr-6.8.0 \
 --type "SPARK" \
 --architecture "ARM64" \
```

```
--region us-west-2
```

EMR Studio

Pour lancer une application à l'aide des processeurs Graviton d'EMR Studio, choisissez arm64 comme option Architecture lorsque vous créez ou mettez à jour une application.

Configuration des applications existantes pour utiliser Graviton

Vous pouvez configurer vos applications Amazon EMR Serverless existantes pour utiliser l'architecture Graviton (arm64) avec le SDK ou EMR Studio. AWS CLI

Pour convertir une application existante de x86 en arm64

1. Vérifiez que vous utilisez la dernière version majeure du AWS CLI/SDK qui prend en charge le `architecture` paramètre.
2. Vérifiez qu'aucune tâche n'est en cours d'exécution, puis arrêtez l'application.

```
aws emr-serverless stop-application \
 --application-id application-id \
 --region us-west-2
```

3. Pour mettre à jour l'application afin d'utiliser Graviton, spécifiez ARM64 le `architecture` paramètre dans l'`update-application`API.

```
aws emr-serverless update-application \
 --application-id application-id \
 --architecture 'ARM64' \
 --region us-west-2
```

4. Pour vérifier que l'architecture du processeur de l'application est correcte ARM64, utilisez l'`get-application`API.

```
aws emr-serverless get-application \
 --application-id application-id \
 --region us-west-2
```

5. Lorsque vous êtes prêt, redémarrez l'application.

```
aws emr-serverless start-application \
```

```
--application-id application-id \
--region us-west-2
```

Considérations relatives à l'utilisation de Graviton

Avant de lancer une application EMR sans serveur utilisant arm64 pour le support de Graviton, vérifiez les points suivants.

Compatibilité avec les bibliothèques

Lorsque vous sélectionnez Graviton (arm64) comme option d'architecture, assurez-vous que les packages et bibliothèques tiers sont compatibles avec l'architecture ARM 64 bits. Pour plus d'informations sur la façon d'empaqueter des bibliothèques Python dans un environnement virtuel Python compatible avec l'architecture que vous avez sélectionnée, reportez-vous àUtilisation de bibliothèques Python avec EMR Serverless.

Pour en savoir plus, consultez le référentiel AWS Graviton Getting Started sur GitHub. Ce référentiel contient des ressources essentielles qui peuvent vous aider à démarrer avec le Graviton basé sur ARM.

Job simultané et mise en file d'attente pour une application EMR sans serveur

À partir de la version 7.0.0 d'Amazon EMR et des versions ultérieures, spécifiez le délai d'expiration de la file d'exécution des tâches et la configuration de simultanéité pour votre application. Lorsque vous spécifiez cette configuration, Amazon EMR Serverless commence par mettre votre tâche en file d'attente et commence son exécution en fonction de l'utilisation simultanée de votre application. Par exemple, si le nombre de tâches exécutées simultanément est de 10, seules dix tâches sont exécutées à la fois sur votre application. Les tâches restantes sont mises en file d'attente jusqu'à ce que l'une des tâches en cours soit terminée. Si le délai d'attente est atteint plus tôt, votre tâche expire. Pour plus d'informations, reportez-vous à la section États d'exécution des tâches.

Principaux avantages de la simultanéité et de la mise en file d'attente

La simultanéité des tâches et la mise en file d'attente offrent les avantages suivants lorsque de nombreuses soumissions d'offres d'emploi sont requises :

- Il permet de contrôler les tâches exécutées simultanément afin d'utiliser efficacement les limites de capacité au niveau de l'application.
- La file d'attente peut contenir une rafale soudaine de soumissions de tâches, avec un paramètre de délai d'expiration configurable.

Commencer à utiliser la simultanéité et les files d'attente

Les procédures suivantes montrent différentes manières de mettre en œuvre la simultanéité et la mise en file d'attente.

En utilisant le AWS CLI

1. Créez une application Amazon EMR Serverless avec un délai d'attente expirant et des exécutions de tâches simultanées :

```
aws emr-serverless create-application \
--release-label emr-7.0.0 \
--type SPARK \
--scheduler-configuration '{"maxConcurrentRuns": 1, "queueTimeoutMinutes": 30}'
```

2. Mettez à jour une application pour modifier le délai d'expiration et la simultanéité de la file d'attente des tâches :

```
aws emr-serverless update-application \
--application-id application-id \
--scheduler-configuration '{"maxConcurrentRuns": 5, "queueTimeoutMinutes": 30}'
```

> **Note**
>
> Vous pouvez mettre à jour votre application existante pour activer la simultanéité des tâches et la mise en file d'attente. Pour ce faire, l'application doit avoir une étiquette de version emr-7.0.0 ou ultérieure.

En utilisant le AWS Management Console

Les étapes suivantes montrent comment démarrer avec la simultanéité des tâches et la mise en file d'attente à l'aide de : AWS Management Console

1. Accédez à EMR Studio et choisissez de créer une application avec le label de version EMR-7.0.0 ou supérieur.
2. Sous Options de configuration de l'application, sélectionnez l'option Utiliser des paramètres personnalisés.
3. Sous Configurations supplémentaires, vous trouverez une section consacrée aux paramètres de Job Run. Sélectionnez l'option Activer la simultanéité des tâches pour activer la fonctionnalité.
4. Après la sélection, sélectionnez Exécutions de tâches simultanées et Délai d'attente pour configurer respectivement le nombre d'exécutions de tâches simultanées et le délai d'expiration de la file d'attente. Si vous ne saisissez aucune valeur pour ces paramètres, les valeurs par défaut sont utilisées.
5. Choisissez Créer une application et l'application sera créée avec cette fonctionnalité activée. Pour vérifier, accédez au tableau de bord, sélectionnez votre application et vérifiez sous l'onglet Propriétés si la fonctionnalité est activée.

Après la configuration, soumettez les tâches avec cette fonctionnalité activée.

Considérations relatives à la simultanéité et à la mise en file d'attente

Tenez compte des éléments suivants lorsque vous implémentez la simultanéité et la mise en file d'attente :

- La simultanéité des tâches et la mise en file d'attente sont prises en charge sur les versions 7.0.0 et supérieures d'Amazon EMR.
- La simultanéité des tâches et la mise en file d'attente sont activées par défaut sur Amazon EMR version 7.3.0 et versions ultérieures.
- Vous ne pouvez pas mettre à jour la simultanéité pour une application à l'état DÉMARRÉ.
- La plage valide `maxConcurrentRuns` est comprise entre 1 et 1 000, et entre 15 et 720. `queueTimeoutMinutes`
- Un maximum de 2 000 emplois peuvent être en attente pour un compte.
- La simultanéité et la mise en file d'attente s'appliquent aux tâches par lots et en streaming. Il ne peut pas être utilisé pour des tâches interactives. Pour plus d'informations, reportez-vous à Exécuter des charges de travail interactives avec EMR Serverless via EMR Studio.

Transférez des données dans S3 Express One Zone avec EMR Serverless

Avec les versions 7.2.0 et supérieures d'Amazon EMR, utilisez EMR Serverless 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

- Autorisations de zone S3 Express One : lorsque S3 Express One Zone effectue initialement une action similaire `GET` ou `PUT` sur un objet S3, la classe de stockage appelle `CreateSession` en votre nom. `LIST` Votre politique IAM doit accorder l'autorisation `s3express:CreateSession` pour que le connecteur S3A puisse invoquer l'API `CreateSession`. Pour un exemple de politique assortie de cette autorisation, reportez-vous àBien démarrer avec S3 Express One Zone.
- S3Aconnecteur — Pour configurer Spark afin d'accéder aux données d'un compartiment Amazon S3 utilisant la classe de stockage S3 Express One Zone, utilisez le connecteur Apache Hadoop. S3A Pour utiliser le connecteur, assurez-vous que tous les S3 URIs utilisent le `s3a` schéma. Si ce n'est pas le cas, modifiez l'implémentation `s3` et les `s3n` schémas du système de fichiers que vous utilisez.

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"
    }
  }
]
```

Bien démarrer avec S3 Express One Zone

Suivez ces étapes pour commencer à utiliser S3 Express One Zone.

1. Créez un point de terminaison VPC. Ajoutez le point de terminaison `com.amazonaws.us-west-2.s3express` au point de terminaison du VPC.
2. Suivez Getting started with Amazon EMR Serverless pour créer une application avec le label de version 7.2.0 ou supérieur d'Amazon EMR.
3. Configurez votre application pour utiliser le point de terminaison VPC nouvellement créé, un groupe de sous-réseaux privés et un groupe de sécurité.
4. Ajoutez l'`CreateSession`autorisation à votre rôle d'exécution des tâches.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "s3express:CreateSession"
      ],
      "Sid": "AllowS3EXPRESSCreatesession"
    }
```

```
  ]
}
```

5. Exécutez votre travail. Notez que vous devez utiliser le `S3A` schéma pour accéder aux compartiments S3 Express One Zone.

```
aws emr-serverless start-job-run \
--application-id <application-id> \
--execution-role-arn <job-role-arn> \
--name <job-run-name> \
--job-driver '{
 "sparkSubmit": {


 "entryPoint": "s3a://<DOC-EXAMPLE-BUCKET>/scripts/wordcount.py",
 "entryPointArguments":["s3a://<DOC-EXAMPLE-BUCKET>/emr-serverless-spark/output"],
 "sparkSubmitParameters": "--conf spark.executor.cores=4
 --conf spark.executor.memory=8g --conf spark.driver.cores=4
 --conf spark.driver.memory=8g --conf spark.executor.instances=2
 --conf spark.hadoop.fs.s3a.change.detection.mode=none
 --conf spark.hadoop.fs.s3a.endpoint.region={<AWS_REGION>}
 --conf spark.hadoop.fs.s3a.select.enabled=false
 --conf spark.sql.sources.fastS3PartitionDiscovery.enabled=false
 }'
```

Exécution de tâches

Après avoir soumis votre candidature, soumettez-y des offres d'emploi. Cette section explique comment utiliser le AWS CLI pour exécuter ces tâches. Cette section identifie également les valeurs par défaut pour chaque type d'application disponible sur EMR Serverless.

Rubriques

États d'exécution de la tâche

Lorsque vous soumettez une tâche exécutée à une file d'attente de tâches Amazon EMR Serverless, la tâche exécutée entre dans l'état. `SUBMITTED` L'état d'une tâche passe de « à » `SUBMITTED` `RUNNING` jusqu'à ce qu'il atteigne `FAILED``SUCCESS`, ou`CANCELLING`.

Les exécutions de tâches peuvent avoir les états suivants :

State	Description
Soumis	État initial de la tâche lorsque vous soumettez une tâche exécutée à EMR Serverless. Le poste attend d'être programmé pour la candidature. EMR Serverless commence à prioriser et à planifier l'exécution de la tâche.
En file d’attente	L'exécution de la tâche attend dans cet état lorsque la simultanéité d'exécution des tâches au niveau de l'application est entièrement occupée. Pour plus d'informations sur la mise en file d'attente et la simultanéité, reportez-vous à. Job simultané et mise en file d'attente pour une application EMR sans serveur
En suspens	Le planificateur évalue l'exécution du travail afin de prioriser et de planifier l'exécution de l'application.
Programmé	EMR Serverless a planifié l'exécution de la tâche pour l'application et alloue des ressource s pour exécuter la tâche.
En cours d'exécution	EMR Serverless a alloué les ressources dont la tâche avait initialement besoin, et la tâche est en cours d'exécution dans l'application. Dans les applications Spark, cela signifie que le processus du pilote Spark est en état `running`.
Échec	EMR Serverless n'a pas réussi à envoyer la tâche exécutée à l'application, ou elle s'est terminée sans succès. Reportez-vous à `StateDetails` pour plus d'informations sur cet échec de tâche.
Succès	L'exécution de la tâche s'est terminée avec succès.

State	Description
Annulation	L'`CancelJobRun` API a demandé l'annulation de l'exécution du travail ou le délai d'exécuti on du travail a expiré. EMR Serverless essaie d'annuler la tâche dans l'application et de libérer les ressources.
Annulée	L'exécution de la tâche a été annulée avec succès et les ressources utilisées ont été libérées.

Annulation de l'exécution d'une tâche EMR sans serveur avec délai de grâce

Dans les systèmes de traitement des données, les interruptions brusques peuvent entraîner un gaspillage de ressources, des opérations incomplètes et des incohérences potentielles dans les données. Amazon EMR Serverless vous permet de définir un délai de grâce lors de l'annulation de l'exécution d'une tâche. Cette fonctionnalité laisse le temps de procéder au nettoyage approprié et de terminer les travaux en cours avant la fin du travail.

Lorsque vous annulez l'exécution d'une tâche, spécifiez un délai de grâce (en secondes) à l'aide du paramètre `shutdownGracePeriodInSeconds` pendant lequel la tâche peut effectuer des opérations de nettoyage avant son arrêt final. Le comportement et les paramètres par défaut varient entre les tâches par lots et les tâches de streaming.

Période de grâce pour les tâches par lots

Pour les tâches par lots, EMR Serverless vous permet de mettre en œuvre des opérations de nettoyage personnalisées qui s'exécutent pendant la période de grâce. Vous pouvez enregistrer ces opérations de nettoyage dans le cadre du hook d'arrêt de la JVM dans le code de votre application.

Comportement par défaut

Le comportement par défaut pour l'arrêt est de n'avoir aucun délai de grâce. Il comprend les deux actions suivantes :

- Résiliation immédiate

- Les ressources sont débloquées immédiatement

Options de configuration

Vous pouvez définir des paramètres qui se traduiront par un arrêt progressif :

- Plage valide pour la période de grâce d'arrêt : 15 à 1800 secondes (facultatif)
- Résiliation immédiate (sans délai de grâce) : 0 seconde

Activer l'arrêt progressif

Pour implémenter un arrêt progressif pour les tâches par lots, procédez comme suit :

1. Ajoutez un hook d'arrêt dans le code de votre application contenant une logique d'arrêt personnalisée.

   Example in Scala

   ```
   import org.apache.hadoop.util.ShutdownHookManager

   // Register shutdown hook with priority (second argument)
   // Higher priority hooks run first
   ShutdownHookManager.get().addShutdownHook(() => {
       logger.info("Performing cleanup operations...")
   }, 100)
   ```

   Utilisation de ShutdownHookManager

   Example in PySpark

   ```
   import atexit

   def cleanup():
       # Your cleanup logic here
       print("Performing cleanup operations...")

   # Register the cleanup function
   atexit.register(cleanup)
   ```

2. Spécifiez un délai de grâce lors de l'annulation de la tâche afin de laisser le temps aux hooks ajoutés précédemment de s'exécuter

Exemple

```
# Default (immediate termination)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID

# With 5-minute grace period
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 300
```

Période de grâce pour les emplois de streaming

Dans Spark Structured Streaming, où les calculs impliquent la lecture ou l'écriture dans des sources de données externes, les arrêts brusques peuvent entraîner des résultats indésirables. Les tâches de streaming traitent les données par microlots, et l'interruption de ces opérations en cours de route peut entraîner un double traitement lors de tentatives ultérieures. Cela se produit lorsque le dernier point de contrôle du microlot précédent n'a pas été écrit, ce qui entraîne le nouveau traitement des mêmes données au redémarrage de la tâche de streaming. Un tel traitement dupliqué gaspille non seulement des ressources informatiques, mais peut également avoir un impact sur les opérations commerciales, d'où la nécessité d'éviter les arrêts brusques.

EMR Serverless fournit un support intégré pour un arrêt progressif via un écouteur de requêtes en streaming. Cela garantit l'achèvement correct des microlots en cours avant la fin du travail. Le service gère automatiquement l'arrêt progressif entre les microlots pour les applications de streaming, en veillant à ce que le microlot en cours termine le traitement, que les points de contrôle soient correctement écrits et que le contexte de diffusion soit correctement arrêté sans ingestion de nouvelles données pendant le processus d'arrêt.

Comportement par défaut

- Période de grâce de 120 secondes activée par défaut.
- L'écouteur de requêtes de streaming intégré gère un arrêt progressif.

Options de configuration

- Plage valide pour la période de grâce d'arrêt : 15 à 1800 secondes (facultatif)
- Fin immédiate : 0 seconde

Activer Graceful Shutdown

Pour implémenter un arrêt progressif pour les tâches de streaming, procédez comme suit :

Spécifiez une période de grâce lors de l'annulation de la tâche afin de laisser le temps au microlot en cours d'être terminé.

Exemple

```
# Default graceful shutdown (120 seconds)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID

# Custom grace period (e.g. 300 seconds)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 300

# Immediate Termination
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 0
```

Ajoutez des crochets d'arrêt personnalisés (facultatif)

Alors qu'EMR Serverless gère l'arrêt progressif par défaut via son écouteur de requêtes de streaming intégré, vous pouvez éventuellement implémenter une logique d'arrêt personnalisée pour les requêtes de streaming individuelles. EMR Serverless enregistre son écouteur d'arrêt progressif avec la priorité 60 (utilisation). ShutdownHookManager Étant donné que les hooks à priorité plus élevée s'exécutent en premier, vous pouvez enregistrer vos opérations de nettoyage personnalisées avec une priorité supérieure à 60 pour vous assurer qu'elles s'exécutent avant le début du processus d'arrêt d'EMR Serverless.

Pour ajouter un hook personnalisé, reportez-vous au premier exemple de cette rubrique qui montre comment ajouter un hook d'arrêt dans le code de votre application. Ici, 100 est la priorité, ce qui est supérieur à 60. Par conséquent, un tel crochet d'arrêt s'exécute en premier.

> **Note**
>
> Les hooks d'arrêt personnalisés sont facultatifs et ne sont pas nécessaires pour la fonctionnalité d'arrêt progressif, qui est gérée automatiquement par EMR Serverless.

Frais liés à la période de grâce et durée du Batch

Si la valeur par défaut du délai de grâce (120 secondes) est utilisée :

- Si la durée de votre lot est inférieure à 120 secondes, vous ne serez facturé que pour le temps réellement nécessaire pour terminer le lot.
- Si la durée de votre lot dépasse 120 secondes, le délai de grâce maximal (120 secondes) vous sera facturé, mais la requête risque de ne pas s'arrêter correctement car elle sera interrompue de force.

Pour optimiser les coûts et garantir un arrêt progressif :

- Pour les durées de lot supérieures à 120 secondes : envisagez d'augmenter le délai de grâce pour qu'il corresponde à la durée de votre lot
- Pour les durées de lot inférieures à 120 secondes : il n'est pas nécessaire d'ajuster le délai de grâce car vous ne serez facturé que pour le temps de traitement réel

Considérations

Comportement du délai de grâce

- La période de grâce vous donne le temps de terminer les interruptions enregistrées.
- Job se termine dès que le crochet d'arrêt est terminé, même si c'est bien avant la période de grâce.
- Si les opérations de nettoyage dépassent le délai de grâce, le travail sera résilié de force.

Comportement du service

- L'arrêt par période de grâce n'est disponible que pour les tâches en cours d'exécution.
- Les demandes d'annulation suivantes pendant l'état CANCELING sont ignorées.
- Si EMR Serverless ne parvient pas à initier l'arrêt du délai de grâce en raison d'erreurs de service internes :
  - Le service réessaiera pendant 2 minutes au maximum.
  - Si les nouvelles tentatives échouent, la tâche sera interrompue de force.

Facturation

Les tâches sont facturées en fonction des ressources informatiques utilisées jusqu'à leur arrêt complet, y compris le temps nécessaire pendant la période de grâce.

Exécution de tâches depuis la console EMR Studio

Vous pouvez envoyer des exécutions de tâches aux applications EMR Serverless et accéder aux tâches depuis la console EMR Studio. Pour créer ou accéder à votre application EMR Serverless sur la console EMR Studio, suivez les instructions de la section Démarrage depuis la console.

Envoi d'une tâche

Sur la page Soumettre une tâche, soumettez une tâche à une application EMR Serverless comme suit.

Spark

1. Dans le champ Nom, saisissez le nom de l'exécution de votre tâche.
2. Dans le champ Rôle d'exécution, entrez le nom du rôle IAM que votre application EMR Serverless peut assumer pour l'exécution de la tâche. Pour en savoir plus sur les rôles d'exécution, reportez-vous àRôles d'exécution des tâches pour Amazon EMR Serverless.
3. Dans le champ Emplacement du script, entrez l'emplacement Amazon S3 du script ou du JAR que vous souhaitez exécuter. Pour les tâches Spark, le script peut être un fichier Python (`.py`) ou un fichier JAR (`.jar`).
4. Si l'emplacement de votre script est un fichier JAR, entrez le nom de classe qui est le point d'entrée de la tâche dans le champ Classe principale.

5. (Facultatif) Entrez des valeurs pour les autres champs.

   - Arguments du script — Entrez les arguments que vous souhaitez transmettre à votre script JAR ou Python principal. Votre code lit ces paramètres. Séparez chaque argument du tableau par une virgule.
   - Propriétés de Spark — Développez la section des propriétés de Spark et entrez les paramètres de configuration de Spark dans ce champ.

     > **Note**
     >
     > Si vous spécifiez la taille du pilote et de l'exécuteur Spark, tenez compte de la surcharge mémoire. Spécifiez les valeurs de surcharge de mémoire dans les propriétés `spark.driver.memory0verhead` et`spark.executor.memory0verhead`. La surcharge mémoire a une valeur par défaut de 10 % de la mémoire du conteneur, avec un minimum de 384 Mo. La mémoire de l'exécuteur et la surcharge de mémoire réunies ne peuvent pas dépasser la mémoire de travail. Par exemple, le maximum `spark.executor.memory` d'un travailleur de 30 Go doit être de 27 Go.

   - Configuration de la tâche — Spécifiez n'importe quelle configuration de tâche dans ce champ. Vous pouvez utiliser ces configurations de tâches pour remplacer les configurations par défaut des applications.
   - Paramètres supplémentaires — Activez ou désactivez le catalogue de données AWS Glue en tant que métastore et modifiez les paramètres du journal des applications. Pour en savoir plus sur les configurations de métastore, reportez-vous à. Configuration du métastore pour EMR Serverless Pour en savoir plus sur les options de journalisation des applications, reportez-vous àStockage des journaux.
   - Balises — Attribuez des balises personnalisées à l'application.

6. Choisissez Soumettre une tâche.

Hive

1. Dans le champ Nom, saisissez le nom de l'exécution de votre tâche.
2. Dans le champ Rôle d'exécution, entrez le nom du rôle IAM que votre application EMR Serverless peut assumer pour l'exécution de la tâche.

3. Dans le champ Emplacement du script, entrez l'emplacement Amazon S3 du script ou du JAR que vous souhaitez exécuter. Pour les jobs Hive, le script doit être un fichier Hive (`.sql`).
4. (Facultatif) Entrez des valeurs pour les autres champs.
   - Emplacement du script d'initialisation : entrez l'emplacement du script qui initialise les tables avant l'exécution du script Hive.
   - Propriétés de la ruche — Développez la section des propriétés de la ruche et entrez les paramètres de configuration de la ruche dans ce champ.
   - Configuration de la tâche : spécifiez n'importe quelle configuration de tâche. Vous pouvez utiliser ces configurations de tâches pour remplacer les configurations par défaut des applications. Pour les tâches Hive, `hive.exec.scratchdir` et `hive.metastore.warehouse.dir` sont des propriétés obligatoires dans la `hive-site` configuration.

     ```
     {
         "applicationConfiguration": [
             {
                 "classification": "hive-site",
                 "configurations": [],
                 "properties": {
                     "hive.exec.scratchdir": "s3://DOC-EXAMPLE_BUCKET/hive/
     scratch",
                     "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE_BUCKET/hive/
     warehouse"
                 }
             }
         ],
         "monitoringConfiguration": {}
     }
     ```

   - Paramètres supplémentaires — Activez ou désactivez le catalogue de données AWS Glue en tant que métastore et modifiez les paramètres du journal des applications. Pour en savoir plus sur les configurations de métastore, reportez-vous à. Configuration du métastore pour EMR Serverless Pour en savoir plus sur les options de journalisation des applications, reportez-vous àStockage des journaux.
   - Balises — Attribuez des balises personnalisées à l'application.
5. Choisissez Soumettre une tâche.

Exécution des tâches d'accès

À partir de l'onglet Exécutions de tâches sur la page Détails d'une application, accédez aux exécutions de tâches et effectuez les actions suivantes pour les exécutions de tâches.

Annuler la tâche : pour annuler l'exécution d'une tâche en l'`RUNNING`état, choisissez cette option. Pour en savoir plus sur les transitions d'exécution de tâches, reportez-vous àÉtats d'exécution de la tâche.

Cloner une tâche : pour cloner une tâche précédente et la soumettre à nouveau, choisissez cette option.

Exécution de tâches à partir du AWS CLI

Vous pouvez créer, décrire et supprimer des tâches individuelles sur le AWS CLI. Vous pouvez également répertorier toutes vos offres d'emploi pour y accéder en un coup d'œil.

Pour soumettre une nouvelle offre d'emploi, utilisez`start-job-run`. Indiquez l'ID de l'application que vous souhaitez exécuter, ainsi que les propriétés spécifiques à la tâche. Pour des exemples de Spark, reportez-vous àUtilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur. Pour des exemples de Hive, reportez-vous à. Utilisation de configurations Hive lorsque vous exécutez des tâches EMR sans serveur Cette commande renvoie votre`application-id`, votre ARN et un nouveau`job-id`.

Chaque tâche exécutée a une durée de temporisation définie. Si la durée d'exécution de la tâche dépasse cette durée, EMR Serverless l'annulera automatiquement. Le délai d'expiration par défaut est de 12 heures. Lorsque vous démarrez l'exécution de votre tâche, configurez ce paramètre de délai d'expiration sur une valeur qui répond aux exigences de votre tâche. Configurez la valeur avec la `executionTimeoutMinutes` propriété.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --execution-timeout-minutes 15 \
  --job-driver '{
    "hive": {
        "query": "s3://amzn-s3-demo-bucket/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/hive/
warehouse"
```

```
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

Pour décrire un travail, utilisez`get-job-run`. Cette commande renvoie les configurations spécifiques à la tâche et la capacité définie pour votre nouvelle tâche.

```
aws emr-serverless get-job-run \
--job-run-id job-id \
--application-id application-id
```

Pour répertorier vos offres d'emploi, utilisez`list-job-runs`. Cette commande renvoie un ensemble abrégé de propriétés qui inclut le type de tâche, l'état et d'autres attributs de haut niveau. Si vous ne souhaitez pas accéder à toutes vos tâches, spécifiez le nombre maximum de tâches auxquelles vous souhaitez accéder, jusqu'à 50. L'exemple suivant indique que vous souhaitez accéder à vos deux dernières exécutions de tâches.

```
aws emr-serverless list-job-runs \
--max-results 2 \
--application-id application-id
```

Pour annuler une tâche, utilisez`cancel-job-run`. Indiquez le `application-id` et le `job-id` nom du travail que vous souhaitez annuler.

```
aws emr-serverless cancel-job-run \
--job-run-id job-id \
--application-id application-id
```

Pour plus d'informations sur la façon d'exécuter des tâches à partir de AWS CLI, reportez-vous au manuel EMR Serverless API Reference.

Politique d'exécution IAM

Vous pouvez spécifier une politique IAM d'exécution, en plus d'un rôle d'exécution, lorsque vous soumettez des exécutions de tâches sur EMR Serverless. Les autorisations obtenues par l'exécution du job correspondent à l'intersection des autorisations du rôle d'exécution et de la politique IAM d'exécution spécifiée.

Démarrage

Étapes pour utiliser la politique d'exécution IAM :

Créez une `emr-serverless` application ou utilisez une application existante, puis exécutez l'interface de ligne de commande aws suivante pour démarrer une tâche avec une politique IAM intégrée :

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options \
      --execution-iam-policy '{"policy": "inline-policy"}'
```

Exemples de commandes CLI

Si la politique suivante est enregistrée dans le `policy.json` fichier sur la machine :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket",
        "arn:aws:s3:::my-test-bucket/*"
      ],
      "Sid": "AllowS3Getobject"
```

```
    }
  ]
}
```

Ensuite, nous pouvons démarrer un travail avec cette politique à l'aide de la AWS CLI commande suivante :

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options
      --execution-iam-policy '{
          "policy": '$(jq -c '. | @json' policy.json)'
      }'
```

Vous pouvez également utiliser les deux politiques AWS et les politiques gérées par le client, en les spécifiant par le biais de leurs ARNs :

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options
      --execution-iam-policy '{
          "policyArns": [
          "arn:aws:iam::aws:policy/AmazonS3FullAccess",
          "arn:aws:iam::aws:policy/CloudWatchLogsFullAccess"
          ]
    }'
```

Il est également possible de spécifier à la fois une stratégie IAM intégrée et une politique gérée ARNs dans la même demande :

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options
      --execution-iam-policy '{
          "policy": '$(jq -c '. | @json' policy.json)',
          "policyArns": [
          "arn:aws:iam::aws:policy/AmazonS3FullAccess",
          "arn:aws:iam::aws:policy/CloudWatchLogsFullAccess"
```

```
        ]
    }'
```

Remarques importantes

- Le `execution-role-policy` `policy` champ s peut avoir une longueur maximale de 2 048 caractères.
- La chaîne de politique IAM en ligne spécifiée dans le `execution-iam-policy` `policy` champ « s » doit être conforme à la norme de chaîne JSON, sans aucune nouvelle ligne ni aucun guillemet, comme dans l'exemple précédent.
- Une liste contenant jusqu'à 10 politiques gérées ARNs peut être spécifiée sous forme de valeur `policyArns` dans `execution-iam-policy` le champ.
- La politique gérée ARNs doit être une liste d'ARN de stratégie valides AWS ou gérés par le client. Lorsqu'un ARN de stratégie géré par le client est spécifié, la politique doit appartenir au même AWS compte de l'EMR-S JobRun.
- Lorsque la stratégie IAM intégrée et les politiques gérées sont utilisées, le texte en clair que vous utilisez pour les politiques intégrées et gérées combinées ne peut pas dépasser 2 048 caractères.
- Les autorisations obtenues par le JobRun sont l'intersection des autorisations du rôle d'exécution et de la politique IAM d'exécution spécifiée.

Intersection des politiques

Les autorisations obtenues par l'exécution du job correspondent à l'intersection des autorisations du rôle d'exécution et de la politique IAM d'exécution spécifiée. Cela signifie que toute autorisation requise devra être spécifiée aux deux endroits pour JobRun pouvoir fonctionner. Cependant, il est possible de spécifier une instruction d'autorisation générale supplémentaire dans la politique intégrée pour toutes les autorisations que vous n'avez pas l'intention de mettre à jour ou de remplacer.

Exemple

Compte tenu de la politique de rôle IAM d'exécution suivante :

JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowS3"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:123456789012:log-group::log-stream"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:*:table/MyCompany1table"
      ],
      "Sid": "AllowDYNAMODBDescribetable"
    }
  ]
}
```

Et la politique IAM intégrée suivante :

JSON

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

```
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket/tenant1",
        "arn:aws:s3:::my-test-bucket/tenant1/*"
      ],
      "Sid": "AllowS3Getobject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:*",
        "dynamodb:*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGS"
    }
  ]
}
```

Les autorisations ainsi obtenues JobRun sont les suivantes :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket/tenant1",
        "arn:aws:s3:::my-test-bucket/tenant1/*"
      ],
      "Sid": "AllowS3Getobject"
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:123456789012:log-group::log-stream"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:*:table/MyCompany1table"
      ],
      "Sid": "AllowDYNAMODBDescribetable"
    }
  ]
}
```

Utilisation de disques optimisés pour le shuffle

Avec les versions 7.1.0 et supérieures d'Amazon EMR, utilisez des disques optimisés pour le shuffle lorsque vous exécutez des tâches Apache Spark ou Hive (afin d'améliorer les performances (I/O-intensive workloads. Compared to standard disks, shuffle-optimized disks provide higher IOPS (I/Oopérations par seconde) afin d'accélérer le mouvement des données et de réduire la latence lors des opérations de shuffle. Les disques optimisés pour le shuffle vous permettent de connecter des disques d'une taille maximale de 2 To par utilisateur. Configurez donc la capacité adaptée aux exigences de votre charge de travail.

Principaux avantages

Les disques optimisés pour le shuffle offrent les avantages suivants.

- Performances IOPS élevées : les disques optimisés pour le shuffle fournissent des IOPS plus élevées que les disques standard, ce qui permet un brassage des données plus efficace et plus

rapide lors des tâches Spark et Hive et d'autres charges de travail intensives en matière de brassage.

- Taille de disque plus importante : les disques optimisés pour le shuffle prennent en charge des tailles de disque allant de 20 Go à 2 To par utilisateur. Choisissez donc la capacité appropriée en fonction de vos charges de travail.

Prise en main

Consultez les étapes suivantes pour utiliser des disques optimisés pour le shuffle dans vos flux de travail.

Spark

1. Créez une application EMR Serverless version 7.1.0 à l'aide de la commande suivante.

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name my-application-name \
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. Configurez votre tâche Spark pour inclure les paramètres `spark.emr-serverless.driver.disk.type` and/or `spark.emr-serverless.executor.disk.type` à exécuter avec des disques optimisés pour le shuffle. Vous pouvez utiliser l'un des paramètres ou les deux, selon votre cas d'utilisation.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi
            --conf spark.executor.cores=4
            --conf spark.executor.memory=20g
            --conf spark.driver.cores=4
            --conf spark.driver.memory=8g
            --conf spark.executor.instances=1
            --conf spark.emr-serverless.executor.disk.type=shuffle_optimized"
```

```
        }
    }'
```

Pour plus d'informations, reportez-vous aux propriétés des tâches Spark.

Hive

1. Créez une application EMR Serverless version 7.1.0 à l'aide de la commande suivante.

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. Configurez votre tâche Hive pour inclure les paramètres à exécuter avec des disques `hive.driver.disk.type` and/or `hive.tez.disk.type` optimisés pour le shuffle. Vous pouvez utiliser l'un des paramètres ou les deux, selon votre cas d'utilisation.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://<DOC-EXAMPLE-BUCKET>/emr-serverless-hive/query/hive-
query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "hive.exec.scratchdir": "s3://<DOC-EXAMPLE-BUCKET>/emr-
serverless-hive/hive/scratch",
                "hive.metastore.warehouse.dir": "s3://<DOC-EXAMPLE-BUCKET>/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1",
                "hive.driver.disk.type": "shuffle_optimized",
```

```
                "hive.tez.disk.type": "shuffle_optimized"
            }
        }]
    }'
```

Pour plus d'informations, consultez les propriétés des tâches Hive.

Configuration d'une application avec une capacité pré-initialisée

Consultez les exemples suivants pour créer des applications basées sur Amazon EMR version 7.1.0. Les propriétés de ces applications sont les suivantes :

- 5 pilotes Spark préinitialisés, chacun doté de 2 vCPU, de 4 Go de mémoire et de 50 Go de disque optimisé pour le shuffle.
- 50 exécuteurs préinitialisés, chacun doté de 4 vCPU, 8 Go de mémoire et 500 Go de disque optimisé pour le shuffle.

Lorsque cette application exécute des tâches Spark, elle consomme d'abord les tâches préinitialisées, puis fait évoluer les tâches à la demande jusqu'à la capacité maximale de 400 vCPU et 1 024 Go de mémoire. Vous pouvez éventuellement omettre la capacité pour l'un `DRIVER` ou `EXECUTOR` l'autre.

Spark

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name <my-application-name> \
  --release-label emr-7.1.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB",
            "disk": "50GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
```

```
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB",
            "disk": "500GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

Hive

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name <my-application-name> \
  --release-label emr-7.1.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB",
            "disk": "50GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB",
            "disk": "500GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

Utilisation du stockage sans serveur pour Amazon EMR Serverless

Avec les versions 7.12 et supérieures d'Amazon EMR, utilisez le stockage sans serveur lorsque vous exécutez des tâches Apache Spark afin d'éliminer le provisionnement de disques locaux, de réduire les coûts de traitement des données et d'éviter les échecs de tâches dus à des contraintes de capacité de disque. Le stockage sans serveur gère automatiquement les opérations de shuffle, de déversement de disque et de mise en cache de disque pour vos tâches sans nécessiter de configuration de capacité et stocke les données intermédiaires gratuitement. Amazon EMR Serverless stocke les données intermédiaires dans un système de stockage sans serveur entièrement géré qui évolue automatiquement en fonction des demandes de charge de travail et permet à Spark de libérer du personnel informatique immédiatement en cas d'inactivité, réduisant ainsi les coûts de calcul.

Principaux avantages

Le stockage sans serveur pour EMR Serverless offre les avantages suivants.

- Stockage sans configuration : le stockage sans serveur élimine le besoin de configurer le type et la taille du disque local pour chaque application ou tâche. EMR Serverless gère automatiquement les opérations de données intermédiaires sans planification des capacités.
- Empêche les échecs de tâches grâce à une mise à l'échelle automatique : la capacité de stockage évolue automatiquement en fonction de la charge de travail, ce qui permet d'éviter les échecs de travail dus à une capacité de disque insuffisante.
- Coûts de traitement des données réduits : le stockage sans serveur réduit les coûts de traitement grâce à deux mécanismes. Tout d'abord, le stockage intermédiaire des données est fourni gratuitement : vous ne payez que pour les ressources de calcul et de mémoire. Deuxièmement, le stockage découplé avec l'allocation dynamique des ressources de Spark permet à Spark de libérer immédiatement les travailleurs lorsqu'ils sont inactifs plutôt que de les conserver pour préserver les données intermédiaires sur les disques locaux. Cela permet une évolutivité et une intégration plus rapides par étape Spark, réduisant ainsi les coûts de calcul pour les tâches où les étapes ultérieures nécessitent moins de travailleurs que les étapes initiales.
- Stockage crypté avec isolation au niveau de la tâche : toutes les données intermédiaires sont cryptées en transit et au repos avec une isolation stricte au niveau de la tâche.
- Prise en charge précise du contrôle d'accès : le stockage sans serveur permet un contrôle d'accès précis grâce à l'intégration d'AWS Lake Formation.

Prise en main

Consultez les étapes suivantes pour utiliser le stockage sans serveur pour EMR Serverless dans vos flux de travail Spark.

1. Création d'une application EMR sans serveur

   Créez une application EMR Serverless version 7.12 (ou ultérieure) avec le stockage sans serveur activé en définissant la propriété spark sur true dans la classification `spark.aws.serverlessStorage.enabled` spark-defaults.

   ```
   aws emr-serverless create-application \
     --type "SPARK" \
     --name my-application \
     --release-label emr-7.12.0 \
     --runtime-configuration '[{
         "classification": "spark-defaults",
           "properties": {
             "spark.aws.serverlessStorage.enabled": "true"
           }
       }]' \
     --region <AWS_REGION>
   ```

2. Démarrer une tâche Spark

   Lancez l'exécution d'une tâche sur votre application. Le stockage sans serveur pour EMR Serverless gère automatiquement les opérations de données intermédiaires telles que le shuffle pour votre travail.

   ```
   aws emr-serverless start-job-run \
     --application-id <application-id> \
     --execution-role-arn <job-role-arn> \
     --job-driver '{
       "sparkSubmit": {
         "entryPoint": "s3://<bucket>/script.py",
         "sparkSubmitParameters": "--conf spark.executor.cores=4
           --conf spark.executor.memory=20g
           --conf spark.driver.cores=4
           --conf spark.driver.memory=8g
           --conf spark.executor.instances=10"
       }
   ```

```
  }'
```

Vous pouvez également activer le stockage sans serveur pour EMR Serverless au niveau de la tâche, même s'il n'est pas activé au niveau de l'application. Cela lancera des nœuds de travail dotés d'un stockage sans serveur pour traiter vos tâches. Vous pouvez également désactiver le stockage sans serveur pour une tâche spécifique en définissant la même propriété Spark `spark.aws.serverlessStorage.enabled` sur false.

```
# Turn on serverless storage for EMR serverless for a specific job
aws emr-serverless start-job-run \
    --application-id <application-id> \
    --execution-role-arn <job-role-arn> \
    --job-driver '{
"sparkSubmit": {
"entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi
            --conf spark.aws.serverlessStorage.enabled": "true"
        }
    }'
```

> **Note**
>
> Pour continuer à utiliser le provisionnement de disque local traditionnel, omettez la **`spark.aws.serverlessStorage.enabled`** configuration ou définissez-la sur false.

Considérations et restrictions

- Version finale : le stockage sans serveur est pris en charge sur Amazon EMR version 7.12 et versions ultérieures.
- Limites de volume de données : chaque tâche peut lire et écrire jusqu'à 200 Go de données intermédiaires par tâche exécutée. Les tâches dépassant cette limite échoueront avec un message d'erreur indiquant que la limite de stockage sans serveur a été atteinte.
- Délai d'exécution des tâches : le stockage sans serveur prend en charge les tâches dont le délai d'exécution peut atteindre 24 heures. Les tâches configurées pour des délais d'exécution plus longs échoueront avec un message d'erreur.

- Capacité préinitialisée : les opérateurs de capacité préinitialisés ne prennent pas en charge le stockage sans serveur. Lorsque vous configurez la capacité préinitialisée, elle ne sera utilisée que par les tâches qui désactivent explicitement le stockage sans serveur au niveau de la tâche. Les tâches pour lesquelles le stockage sans serveur est activé fourniront toujours de nouveaux travailleurs à la demande et n'utiliseront aucune capacité préinitialisée, quelle que soit la configuration au niveau de l'application.
- Types de charge de travail : le stockage sans serveur n'est pas pris en charge pour le streaming et les tâches interactives.
- Configuration du serveur : le stockage sans serveur n'est pas pris en charge pour les utilisateurs utilisant 1 ou 2 v. CPUs

Soutenu Régions AWS

EMR Serverless prend en charge le stockage sans serveur dans les régions suivantes :

- USA Est (Virginie du Nord)
- USA Ouest (Oregon)
- Europe (Irlande)

Tâches de streaming pour le traitement de données diffusées en continu

Une tâche de streaming dans EMR Serverless est un mode de travail qui vous permet d'analyser et de traiter des données de streaming en temps quasi réel. Ces tâches de longue durée interrogent les données de streaming et traitent les résultats en continu au fur et à mesure de leur arrivée. Les tâches de streaming sont particulièrement adaptées aux tâches qui nécessitent un traitement des données en temps réel, telles que les analyses en temps quasi réel, la détection des fraudes et les moteurs de recommandations. Les tâches de streaming EMR sans serveur offrent des optimisations, telles que la résilience intégrée des tâches, une surveillance en temps réel, une gestion améliorée des journaux et l'intégration avec des connecteurs de streaming.

Voici quelques exemples d'utilisation des jobs de streaming :

- Analyses en temps quasi réel : les tâches de streaming dans Amazon EMR Serverless vous permettent de traiter les données de streaming en temps quasi réel. Vous pouvez ainsi effectuer des analyses en temps réel sur des flux de données continus, tels que les données de journal, les

données de capteurs ou les données de parcours de navigation afin d'en tirer des informations et de prendre des décisions en temps opportun sur la base des informations les plus récentes.

- Détection des fraudes : utilisez les tâches de streaming pour détecter les fraudes en temps quasi réel dans les transactions financières, les opérations de carte de crédit ou les activités en ligne lorsque vous analysez des flux de données et identifiez des modèles ou des anomalies suspects au fur et à mesure qu'ils se produisent.
- Moteurs de recommandation : les tâches de streaming peuvent traiter les données d'activité des utilisateurs et mettre à jour les modèles de recommandations. Cela ouvre la voie à des recommandations personnalisées et en temps réel basées sur les comportements et les préférences.
- Analyse des réseaux sociaux : les jobs de streaming peuvent traiter les données des réseaux sociaux, telles que les tweets, les commentaires et les publications, afin que les entreprises puissent suivre les tendances, analyser les sentiments et gérer la réputation de la marque en temps quasi réel.
- Analyse de l'Internet des objets (IoT) : les tâches de streaming peuvent gérer et analyser des flux de données à haut débit provenant d'appareils IoT, de capteurs et de machines connectées. Ils peuvent donc exécuter la détection des anomalies, la maintenance prédictive et d'autres cas d'utilisation de l'analytique IoT.
- Analyse du flux de clics : les tâches de streaming peuvent traiter et analyser les données du flux de clics provenant de sites Web ou d'applications mobiles. Les entreprises qui utilisent de telles données peuvent effectuer des analyses pour en savoir plus sur le comportement des utilisateurs, personnaliser les expériences utilisateur et optimiser les campagnes marketing.
- Surveillance et analyse des journaux : les tâches de streaming peuvent également traiter les données des journaux provenant de serveurs, d'applications et de périphériques réseau. Cela vous permet de détecter les anomalies, de résoudre les problèmes, ainsi que de garantir l'état et les performances du système.

Principaux avantages

Les tâches de streaming dans EMR Serverless garantissent automatiquement la résilience des tâches, qui est une combinaison des facteurs suivants :

- Réessai automatique : EMR Serverless réessaie automatiquement toutes les tâches qui ont échoué sans aucune intervention manuelle de votre part.
- Résilience de la zone de disponibilité (AZ) : EMR Serverless fait automatiquement passer les tâches de streaming à une AZ saine si l'AZ d'origine rencontre des problèmes.

- Gestion des journaux :
  - Rotation des journaux : pour une gestion plus efficace du stockage sur disque, EMR Serverless effectue régulièrement une rotation des journaux pour les tâches de streaming de longue durée. Cela permet d'éviter l'accumulation de journaux qui pourrait consommer tout l'espace disque.
  - Compaction des journaux : vous aide à gérer et à optimiser efficacement les fichiers journaux dans le cadre d'une persistance gérée. Le compactage améliore également l'expérience de débogage lorsque vous utilisez le serveur d'historique Spark géré.

Sources de données et récepteurs de données pris en charge

EMR Serverless fonctionne avec un certain nombre de sources de données d'entrée et de récepteurs de données de sortie :

- Sources de données d'entrée prises en charge : Amazon Kinesis Data Streams, Amazon Managed Streaming for Apache Kafka et clusters Apache Kafka autogérés. Par défaut, les versions 7.1.0 et supérieures d'Amazon EMR incluent le connecteur Amazon Kinesis Data Streams. Vous n'avez donc pas besoin de créer ou de télécharger de packages supplémentaires.
- Récepteurs de données de sortie pris en charge : tables AWS Glue Data Catalog, Amazon S3, Amazon Redshift, MySQL, PostgreSQL Oracle, Oracle, Microsoft SQL, Apache Iceberg, Delta Lake et Apache Hudi.

Considérations et restrictions

Lorsque vous utilisez des jobs de streaming, gardez à l'esprit les considérations et limites suivantes.

- Les jobs de streaming sont pris en charge par les versions 7.1.0 et supérieures d'Amazon EMR.
- EMR Serverless s'attend à ce que les tâches de streaming s'exécutent pendant une longue période. Vous ne pouvez donc pas définir un délai d'exécution pour limiter le temps d'exécution de la tâche.
- Les jobs de streaming ne sont compatibles qu'avec le moteur Spark, qui repose sur le framework de streaming structuré.
- EMR Serverless réessaie indéfiniment des tâches de streaming, et vous ne pouvez pas personnaliser le nombre maximum de tentatives. La prévention du thrash est automatiquement incluse pour arrêter la nouvelle tentative si le nombre de tentatives infructueuses a dépassé un seuil défini sur une fenêtre horaire. Le seuil par défaut est de cinq tentatives infructueuses en une

heure. Vous pouvez configurer ce seuil pour qu'il soit compris entre 1 et 10 tentatives. Pour plus d'informations, reportez-vous à la section Résilience du travail.

- Les tâches de streaming comportent des points de contrôle pour enregistrer l'état d'exécution et la progression, afin qu'EMR Serverless puisse reprendre la tâche de streaming à partir du dernier point de contrôle. Pour plus d'informations, reportez-vous à la section Restaurer après un échec avec le point de contrôle dans la documentation d'Apache Spark.

Commencer à diffuser des jobs

Consultez les instructions suivantes pour savoir comment démarrer avec les jobs de streaming.

1. Suivez Getting started with Amazon EMR Serverless pour créer une application. Notez que votre application doit exécuter Amazon EMR version 7.1.0 ou supérieure.
2. Une fois que votre application est prête, définissez le `mode` paramètre `STREAMING` sur pour soumettre une tâche de streaming, comme dans l' AWS CLI exemple suivant.

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<streaming script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
            --conf spark.executor.memory=16g
            --conf spark.driver.cores=4
            --conf spark.driver.memory=16g
            --conf spark.executor.instances=3"
    }
}'
```

Connecteurs de streaming pris en charge

Les connecteurs de streaming facilitent la lecture des données depuis une source de streaming et peuvent également écrire des données sur un récepteur de streaming.

Les connecteurs de streaming pris en charge sont les suivants :

Connecteur Amazon Kinesis Data Streams

Le connecteur Amazon Kinesis Data Streams pour Apache Spark permet de créer des applications de streaming et des pipelines qui consomment des données depuis Amazon Kinesis Data Streams et y écrivent des données. Le connecteur prend en charge une consommation de ventilateur améliorée avec un débit de lecture dédié pouvant atteindre 2 Mo/seconde par partition. Par défaut, Amazon EMR Serverless 7.1.0 et versions ultérieures inclut le connecteur. Vous n'avez donc pas besoin de créer ou de télécharger de packages supplémentaires. Pour plus d'informations sur le connecteur, reportez-vous à la spark-sql-kinesis-connectorpage sur GitHub.

L'exemple suivant montre comment démarrer une tâche avec la dépendance du connecteur Kinesis Data Streams.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kinesis-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --jars /usr/share/aws/kinesis/spark-sql-kinesis/lib/spark-streaming-
sql-kinesis-connector.jar"
    }
}'
```

Pour vous connecter à Kinesis Data Streams, configurez l'application EMR Serverless avec un accès VPC et utilisez un point de terminaison VPC pour autoriser l'accès privé, ou utilisez une passerelle NAT pour obtenir un accès public. Pour plus d'informations, reportez-vous à la section Configuration de l'accès au VPC. Vous devez également vous assurer que votre rôle d'exécution des tâches dispose des autorisations de lecture et d'écriture nécessaires pour accéder aux flux de données requis. Pour en savoir plus sur la configuration d'un rôle d'exécution de tâche, consultez la section Rôles d'exécution de tâches pour Amazon EMR Serverless. Pour une liste complète de toutes les autorisations requises, consultez la spark-sql-kinesis-connector page sur GitHub.

Connecteur Apache Kafka

Le connecteur Apache Kafka pour le streaming structuré Spark est un connecteur open source créé par la communauté Spark et est disponible dans un référentiel Maven. Ce connecteur permet aux applications de streaming structuré Spark de lire et d'écrire des données depuis Apache Kafka autogéré et Amazon Managed Streaming for Apache Kafka. Pour plus d'informations sur le connecteur, reportez-vous au guide d'intégration de Structured Streaming + Kafka dans la documentation d'Apache Spark.

L'exemple suivant montre comment inclure le connecteur Kafka dans votre demande d'exécution de tâche.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kafka-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>"
    }
}'
```

La version du connecteur Apache Kafka dépend de la version de votre EMR Serverless et de la version de Spark correspondante. Pour trouver la bonne version de Kafka, reportez-vous au Guide d'intégration de Structured Streaming + Kafka.

Pour utiliser Amazon Managed Streaming for Apache Kafka avec l'authentification IAM, incluez une autre dépendance pour permettre au connecteur Kafka de se connecter à Amazon MSK avec IAM. Pour plus d'informations, consultez le aws-msk-iam-auth référentiel sur GitHub. Vous devez également vous assurer que le rôle d'exécution des tâches dispose des autorisations IAM nécessaires. L'exemple suivant montre comment utiliser le connecteur avec l'authentification IAM.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
```

```
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kafka-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>,software.amazon.msk:aws-msk-iam-
auth:<MSK_IAM_LIB_VERSION>"
    }
}'
```

Pour utiliser le connecteur Kafka et la bibliothèque d'authentification IAM d'Amazon MSK, configurez l'application EMR Serverless avec un accès VPC. Vos sous-réseaux doivent avoir accès à Internet et utiliser une passerelle NAT pour accéder aux dépendances Maven. Pour plus d'informations, reportez-vous à la section Configuration de l'accès au VPC. Les sous-réseaux doivent disposer d'une connectivité réseau pour accéder au cluster Kafka. Cela est vrai que votre cluster Kafka soit autogéré ou que vous utilisiez Amazon Managed Streaming pour Apache Kafka Kafka.

Gestion des journaux de tâches en streaming

Les tâches de streaming prennent en charge la rotation des journaux pour les journaux des applications Spark et les journaux d'événements, ainsi que le compactage des journaux pour les journaux d'événements Spark. Cela vous permet de gérer efficacement vos ressources.

Rotation du journal

Les tâches de streaming prennent en charge la rotation des journaux des applications Spark et des journaux d'événements. La rotation des journaux empêche les longues tâches de streaming de générer des fichiers journaux volumineux susceptibles d'occuper tout l'espace disque disponible. La rotation des journaux vous permet d'économiser de l'espace disque et d'éviter les échecs de tâches dus à un espace disque insuffisant. Pour plus d'informations, reportez-vous à Rotation des journaux.

Compactage des bûches

Les tâches de streaming prennent également en charge le compactage des journaux d'événements Spark chaque fois que la journalisation gérée est disponible. Pour plus de détails sur la journalisation

gérée, reportez-vous à la section Journalisation avec stockage géré. Les tâches de streaming peuvent s'exécuter pendant une longue période, et la quantité de données d'événements peut s'accumuler au fil du temps et augmenter considérablement la taille des fichiers journaux. Le serveur d'historique Spark lit et charge ces événements en mémoire pour l'interface utilisateur de l'application Spark. Ce processus peut entraîner des latences et des coûts élevés, en particulier si les journaux d'événements stockés dans Amazon S3 sont très volumineux.

Le compactage des journaux réduit la taille du journal des événements, de sorte que le serveur d'historique Spark n'a pas besoin de charger plus de 1 Go de journaux d'événements à tout moment. Pour plus d'informations, reportez-vous à la section Surveillance et instrumentation de la documentation d'Apache Spark.

Utilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur

Vous pouvez exécuter des tâches Spark sur une application dont le `type` paramètre est défini sur`SPARK`. Les tâches doivent être compatibles avec la version de Spark compatible avec la version de lancement d'Amazon EMR. Par exemple, lorsque vous exécutez des tâches avec Amazon EMR version 6.6.0, celles-ci doivent être compatibles avec Apache Spark 3.2.0. Pour plus d'informations sur les versions de l'application pour chaque version, reportez-vous àVersions de lancement d'Amazon EMR Serverless.

Paramètres de la tâche Spark

Lorsque vous utilisez l'`StartJobRun`API pour exécuter une tâche Spark, spécifiez les paramètres suivants.

Paramètres requis

- Rôle d'exécution de la tâche Spark
- Paramètre du pilote de tâche Spark
- Paramètre de remplacement de la configuration Spark
- Optimisation dynamique de l'allocation des ressources

Rôle d'exécution de la tâche Spark

Permet **`executionRoleArn`**de spécifier l'ARN du rôle IAM que votre application utilise pour exécuter les tâches Spark. Ce rôle doit contenir les autorisations suivantes :

- Lisez à partir de compartiments S3 ou d'autres sources de données où résident vos données
- Lisez à partir de compartiments ou de préfixes S3 où réside votre PySpark script ou votre fichier JAR
- Écrivez dans les compartiments S3 où vous avez l'intention d'écrire votre sortie finale
- Écrire des journaux dans un compartiment ou un préfixe S3 qui spécifie `S3MonitoringConfiguration`
- Accès aux clés KMS si vous utilisez des clés KMS pour chiffrer les données de votre compartiment S3
- Accès au catalogue de données AWS Glue si vous utilisez SparkSQL

Si votre tâche Spark lit ou écrit des données depuis ou vers d'autres sources de données, spécifiez les autorisations appropriées dans ce rôle IAM. Si vous ne fournissez pas ces autorisations au rôle IAM, la tâche risque d'échouer. Pour plus d'informations, consultez Rôles d'exécution des tâches pour Amazon EMR Serverless et Stockage des journaux.

Paramètre du pilote de tâche Spark

`jobDriver`À utiliser pour fournir des informations sur la tâche. Le paramètre du pilote de tâche n'accepte qu'une seule valeur pour le type de tâche que vous souhaitez exécuter. Pour une tâche Spark, la valeur du paramètre est`sparkSubmit`. Vous pouvez utiliser ce type de tâche pour exécuter Scala PySpark, Java et toute autre tâche prise en charge via Spark Submit. Les tâches Spark ont les paramètres suivants :

- **`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, telles que celles définies dans les arguments `--conf` or`--class`.
- **`entryPointArguments`**— Il s'agit d'un tableau d'arguments que vous souhaitez transmettre à votre fichier JAR ou Python principal. Vous devez gérer la lecture de ces paramètres à l'aide de votre code entrypoint. Séparez chaque argument du tableau par une virgule.
- **`entryPoint`**— Il s'agit de la référence dans Amazon S3 au fichier JAR ou Python principal que vous souhaitez exécuter. Si vous utilisez un fichier JAR Scala ou Java, spécifiez la classe d'entrée principale dans l'`--class`argument `SparkSubmitParameters` using the.

Pour plus d'informations, reportez-vous à la section Lancement d'applications avec spark-submit.

Paramètre de remplacement de la configuration Spark

`configurationOverrides`À utiliser pour remplacer les propriétés de configuration au niveau de la surveillance et au niveau de l'application. Ce paramètre accepte un objet JSON avec les deux champs suivants :

- **`monitoringConfiguration`**- Utilisez ce champ pour spécifier l'URL Amazon S3 (`s3MonitoringConfiguration`) où vous souhaitez que la tâche EMR Serverless stocke les journaux de votre tâche Spark. Assurez-vous d'avoir créé ce compartiment dans le même compartiment Compte AWS que celui qui héberge votre application et dans celui Région AWS où votre tâche est exécutée.
- **`applicationConfiguration`**— Pour remplacer les configurations par défaut des applications, vous pouvez fournir un objet de configuration dans ce champ. 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.

> **Note**
>
> Les classifications de configuration disponibles varient en fonction de la version EMR Serverless spécifique. Par exemple, les classifications pour Log4j personnalisées ne `spark-executor-log4j2` sont disponibles qu'avec les versions 6.8.0 `spark-driver-log4j2` et supérieures.

Si vous utilisez la même configuration dans une dérogation d'application et dans les paramètres d'envoi de Spark, les paramètres d'envoi de Spark sont prioritaires. Les configurations sont classées par ordre de priorité comme suit, du plus haut au plus bas :

- Configuration fournie par EMR Serverless lors de sa création. `SparkSession`
- Configuration que vous fournissez dans le cadre `sparkSubmitParameters` de l'`--conf`argument.
- La configuration que vous fournissez dans le cadre de votre application remplace lorsque vous démarrez une tâche.
- Configuration que vous fournissez `runtimeConfiguration` lors de la création d'une application.

- Configurations optimisées utilisées par Amazon EMR pour cette version.
- Configurations open-source par défaut pour l'application.

Pour plus d'informations sur la déclaration de configurations au niveau de l'application et sur le remplacement de configurations lors de l'exécution d'une tâche, reportez-vous àConfiguration d'application par défaut pour EMR Serverless.

Optimisation dynamique de l'allocation des ressources

`dynamicAllocationOptimization`À utiliser pour optimiser l'utilisation des ressources dans EMR Serverless. La définition de cette propriété sur `true` dans votre classification de configuration Spark indique à EMR Serverless d'optimiser l'allocation des ressources de l'exécuteur afin de mieux aligner le taux auquel Spark demande et annule des exécuteurs avec le taux auquel EMR Serverless crée et libère des travailleurs. EMR Serverless réutilise ainsi de manière plus optimale les travailleurs d'une étape à l'autre, ce qui permet de réduire les coûts lors de l'exécution de tâches comportant plusieurs étapes tout en maintenant les mêmes performances.

Cette propriété est disponible dans toutes les versions d'Amazon EMR.

Voici un exemple de classification de configuration avec`dynamicAllocationOptimization`.

```
[
  {
    "Classification": "spark",
    "Properties": {
      "dynamicAllocationOptimization": "true"
    }
  }
]
```

Tenez compte des points suivants si vous utilisez l'optimisation de l'allocation dynamique :

- Cette optimisation est disponible pour les tâches Spark pour lesquelles vous avez activé l'allocation dynamique des ressources.
- Pour obtenir le meilleur rapport coût-efficacité, nous vous suggérons de configurer une limite d'échelle supérieure pour les travailleurs en utilisant soit le paramètre au niveau de la tâche, soit le paramètre `spark.dynamicAllocation.maxExecutors` de capacité maximale au niveau de l'application en fonction de votre charge de travail.

- Vous ne remarquerez peut-être pas d'amélioration des coûts dans le cas de tâches plus simples. Par exemple, si votre tâche s'exécute sur un petit ensemble de données ou s'exécute en une seule étape, Spark n'a peut-être pas besoin d'un plus grand nombre d'exécuteurs ou de plusieurs événements de dimensionnement.
- Les tâches comportant une séquence composée d'une grande étape, d'étapes plus petites, puis d'une nouvelle étape de grande envergure peuvent connaître une régression au cours de l'exécution des tâches. Dans la mesure où EMR Serverless utilise les ressources de manière plus efficace, cela peut entraîner une diminution du nombre de travailleurs disponibles pour les étapes plus importantes, ce qui se traduit par une durée d'exécution plus longue.

Propriétés de la tâche Spark

Le tableau suivant répertorie les propriétés Spark facultatives et leurs valeurs par défaut que vous pouvez remplacer lorsque vous soumettez une tâche Spark.

Propriétés Spark facultatives et valeurs par défaut

Clé	Description	Valeur par défaut
`spark.archives`	Liste d'archives séparées par des virgules que Spark extrait dans le répertoire de travail de chaque exécuteur . Les types de fichiers pris en charge incluent `.jar` `.tar.gz`, `.tgz` et `.zip`. Pour spécifier le nom du répertoire à extraire, ajoutez-le # après le nom du fichier que vous souhaitez extraire. Par exemple, `file.zip#directory` .	`NULL`
`spark.authenticate`	Option qui active l'authent ification des connexions internes de Spark.	`TRUE`

Clé	Description	Valeur par défaut
`spark.driver.cores`	Nombre de cœurs utilisés par le pilote.	4
`spark.driver.extraJavaOptions`	Options Java supplémentaires pour le pilote Spark.	`NULL`
`spark.driver.memory`	La quantité de mémoire utilisée par le pilote.	14 G
`spark.dynamicAllocation.enabled`	Option qui active l'allocation dynamique des ressources. Cette option augmente ou diminue le nombre d'exécuteurs enregistrés auprès de l'application, en fonction de la charge de travail.	`TRUE`
`spark.dynamicAllocation.executorIdleTimeout`	Durée pendant laquelle un exécuteur peut rester inactif avant que Spark ne le supprime. Cela ne s'applique que si vous activez l'allocation dynamique.	Années 60
`spark.dynamicAllocation.initialExecutors`	Le nombre initial d'exécuteurs à exécuter si vous activez l'allocation dynamique.	`3`
`spark.dynamicAllocation.maxExecutors`	Limite supérieure du nombre d'exécuteurs si vous activez l'allocation dynamique.	Pour les versions 6.10.0 et supérieures, `infinity` Pour les versions 6.9.0 et antérieures, `100`

Clé	Description	Valeur par défaut
`spark.dynamicAllocation.minExecutors`	Limite inférieure du nombre d'exécuteurs si vous activez l'allocation dynamique.	`0`
`spark.emr-serverless.allocation.batch.size`	Le nombre de conteneurs à demander dans chaque cycle d'allocation d'exécuteur. Il y a un intervalle d'une seconde entre chaque cycle d'allocation.	20
`spark.emr-serverless.driver.disk`	Le disque du pilote Spark.	20G
`spark.emr-serverless.driverEnv.` *`[KEY]`*	Option qui ajoute des variables d'environnement au pilote Spark.	NULL
`spark.emr-serverless.executor.disk`	Le disque exécuteur Spark.	20G
`spark.emr-serverless.memoryOverheadFactor`	Définit la surcharge de mémoire à ajouter à la mémoire du conteneur du pilote et de l'exécuteur.	0.1
`spark.emr-serverless.driver.disk.type`	Type de disque connecté au pilote Spark.	Standard
`spark.emr-serverless.executor.disk.type`	Type de disque attaché aux exécuteurs Spark.	Standard
`spark.executor.cores`	Le nombre de cœurs utilisés par chaque exécuteur.	4

Clé	Description	Valeur par défaut
`spark.executor.extraJavaOptions`	Options Java supplémentaires pour l'exécuteur Spark.	`NULL`
`spark.executor.instances`	Nombre de conteneurs d'exécuteurs Spark à allouer.	3
`spark.executor.memory`	La quantité de mémoire utilisée par chaque exécuteur.	14 G
`spark.executorEnv.`*`[KEY]`*	Option qui ajoute des variables d'environnement aux exécuteurs Spark.	`NULL`
`spark.files`	Une liste de fichiers séparés par des virgules à placer dans le répertoire de travail de chaque exécuteur. Vous pouvez accéder aux chemins de ces fichiers dans l'exécuteur avec`SparkFiles.get(`*`fileName`*`)`.	`NULL`
`spark.hadoop.hive.metastore.client.factory.class`	La classe d'implémentation du métastore Hive.	`NULL`
`spark.jars`	JAR supplémentaires à ajouter au chemin de classe d'exécution du pilote et des exécuteurs.	`NULL`
`spark.network.crypto.enabled`	Option qui active le chiffrement RPC basé sur AES. Cela inclut le protocole d'authentification ajouté dans Spark 2.2.0.	`FALSE`

Clé	Description	Valeur par défaut
`spark.sql.warehouse.dir`	Emplacement par défaut pour les bases de données et les tables gérées.	Pour `$PWD/spark-warehouse`
`spark.submit.pyFiles`	Liste séparée par des `.zip` virgules `.egg` de ou de `.py` fichiers à placer dans les applications `PYTHONPATH` Python.	`NULL`

Le tableau suivant répertorie les paramètres d'envoi par défaut de Spark.

Paramètres d'envoi par défaut de Spark

Clé	Description	Valeur par défaut
`archives`	Liste d'archives séparées par des virgules que Spark extrait dans le répertoire de travail de chaque exécuteur.	`NULL`
`class`	Classe principale de l'application (pour les applications Java et Scala).	`NULL`
`conf`	Propriété de configuration arbitraire de Spark.	`NULL`
`driver-cores`	Nombre de cœurs utilisés par le pilote.	4
`driver-memory`	La quantité de mémoire utilisée par le pilote.	14 G
`executor-cores`	Le nombre de cœurs utilisés par chaque exécuteur.	4

Clé	Description	Valeur par défaut
`executor-memory`	La quantité de mémoire utilisée par l'exécuteur.	14 G
`files`	Liste de fichiers séparés par des virgules à placer dans le répertoire de travail de chaque exécuteur. Vous pouvez accéder aux chemins de ces fichiers dans l'exécuteur avec`SparkFiles.get( fileName)`.	`NULL`
`jars`	Une liste de fichiers jar séparés par des virgules à inclure dans les chemins de classe du pilote et de l'exécuteur.	`NULL`
`num-executors`	Le nombre d'exécuteurs à lancer.	3
`py-files`	Une liste séparée par des virgules de `.zip`.`egg`, ou de `.py` fichiers à placer sur les applications `PYTHONPATH` Python.	`NULL`
`verbose`	Option qui active une sortie de débogage supplémentaire.	`NULL`

Bonnes pratiques en matière de configuration des ressources

Configuration des ressources du pilote et de l'exécuteur via l'API StartJobRun

> **Note**
>
> Les cœurs du pilote et de l'exécuteur Spark ainsi que les propriétés de mémoire, s'ils sont spécifiés, doivent être spécifiés directement dans la demande StartJobRun d'API.

La configuration de vos ressources de cette manière garantit qu'EMR Serverless peut allouer les bonnes ressources avant d'exécuter la tâche. Cela contraste avec les paramètres fournis dans le script utilisateur, tels que dans le fichier .py ou .jar, qui sont évalués trop tard, car les pilotes et les exécuteurs sont parfois préconfigurés avant le début de l'exécution du script. Deux méthodes sont prises en charge pour configurer ces ressources lors de la soumission des tâches :

Option 1 : utilisation sparkSubmitParameters

```
"jobDriver": {
 "sparkSubmit": {
    "entryPoint": "s3://your-script-path.py",
    "sparkSubmitParameters": "–conf spark.driver.memory=4g \
    –conf spark.driver.cores=2 \
    –conf spark.executor.memory=8g \
    –conf spark.executor.cores=4"
  }
 }
```

Option 2 : utiliser ConfigurationOverrides pour la classification spark-defaults

```
"configurationOverrides": {
 "applicationConfiguration": [
 {
 "classification": "spark-defaults",
 "properties": {
     "spark.driver.memory": "4g",
     "spark.driver.cores": "2",
     "spark.executor.memory": "8g",
     "spark.executor.cores": "4"
      }
```

```
    }
  ]
 }
```

Exemples Spark

L'exemple suivant montre comment utiliser l'`StartJobRun`API pour exécuter un script Python. Pour un end-to-end didacticiel utilisant cet exemple, reportez-vous àCommencer à utiliser Amazon EMR Serverless. Vous trouverez d'autres exemples d'exécution de PySpark tâches et d'ajout de dépendances Python dans le référentiel EMR Serverless Samples. GitHub

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
        }
    }'
```

L'exemple suivant montre comment utiliser l'`StartJobRun`API pour exécuter un Spark JAR.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
    }'
```

Utilisation de configurations Hive lorsque vous exécutez des tâches EMR sans serveur

Vous pouvez exécuter des tâches Hive sur une application dont le `type` paramètre est défini sur. `HIVE` Les tâches doivent être compatibles avec la version de Hive compatible avec la version de lancement d'Amazon EMR. Par exemple, lorsque vous exécutez des tâches sur une application avec Amazon EMR version 6.6.0, votre tâche doit être compatible avec Apache Hive 3.1.2. Pour plus d'informations sur les versions de l'application pour chaque version, reportez-vous àVersions de lancement d'Amazon EMR Serverless.

Paramètres du job Hive

Lorsque vous utilisez l'`StartJobRun`API pour exécuter une tâche Hive, spécifiez les paramètres suivants.

Paramètres requis

- Rôle d'exécution du job Hive
- Paramètre du pilote de tâche Hive
- Paramètre de remplacement de la configuration Hive

Rôle d'exécution du job Hive

Permet **`executionRoleArn`**de spécifier l'ARN du rôle IAM que votre application utilise pour exécuter les tâches Hive. Ce rôle doit contenir les autorisations suivantes :

- Lisez à partir de compartiments S3 ou d'autres sources de données où résident vos données
- Lisez à partir de compartiments ou de préfixes S3 où résident votre fichier de requête Hive et votre fichier de requête d'initialisation
- Lisez et écrivez dans des compartiments S3 où se trouvent votre répertoire Hive Scratch et votre répertoire d'entrepôt Hive Metastore
- Écrivez dans les compartiments S3 où vous avez l'intention d'écrire votre sortie finale
- Écrire des journaux dans un compartiment ou un préfixe S3 qui spécifie `S3MonitoringConfiguration`
- Accès aux clés KMS si vous utilisez des clés KMS pour chiffrer les données de votre compartiment S3

- Accès au catalogue de données AWS Glue

Si votre tâche Hive lit ou écrit des données depuis ou vers d'autres sources de données, spécifiez les autorisations appropriées dans ce rôle IAM. Si vous ne fournissez pas ces autorisations au rôle IAM, votre tâche risque d'échouer. Pour plus d'informations, consultez Rôles d'exécution des tâches pour Amazon EMR Serverless.

Paramètre du pilote de tâche Hive

`jobDriver`À utiliser pour fournir des informations sur la tâche. Le paramètre du pilote de tâche n'accepte qu'une seule valeur pour le type de tâche que vous souhaitez exécuter. Lorsque vous spécifiez `hive` le type de tâche, EMR Serverless transmet une requête Hive au paramètre. `jobDriver` Les jobs Hive possèdent les paramètres suivants :

- **`query`**— Il s'agit de la référence dans Amazon S3 au fichier de requête Hive que vous souhaitez exécuter.
- **`parameters`**— Il s'agit des propriétés de configuration supplémentaires de Hive que vous souhaitez remplacer. Pour remplacer les propriétés, transmettez-les à ce paramètre en tant que`--hiveconf` *`property=value`*. Pour remplacer des variables, transmettez-les à ce paramètre sous `--hivevar` *`key=value`* la forme.
- **`initQueryFile`**— Il s'agit du fichier de requête init Hive. Hive exécute ce fichier avant votre requête et peut l'utiliser pour initialiser des tables.

Paramètre de remplacement de la configuration Hive

`configurationOverrides`À utiliser pour remplacer les propriétés de configuration au niveau de la surveillance et au niveau de l'application. Ce paramètre accepte un objet JSON avec les deux champs suivants :

- **`monitoringConfiguration`**— Utilisez ce champ pour spécifier l'URL Amazon S3 (`s3MonitoringConfiguration`) où vous souhaitez que la tâche EMR Serverless stocke les journaux de votre tâche Hive. Assurez-vous de créer ce compartiment dans le même compartiment Compte AWS que celui qui héberge votre application et dans celui Région AWS où votre tâche est exécutée.
- **`applicationConfiguration`**— Vous pouvez fournir un objet de configuration dans ce champ pour remplacer les configurations par défaut des applications. 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.

> **Note**
>
> Les classifications de configuration disponibles varient en fonction de la version EMR Serverless spécifique. Par exemple, les classifications pour Log4j personnalisées ne `spark-executor-log4j2` sont disponibles qu'avec les versions 6.8.0 `spark-driver-log4j2` et supérieures.

Si vous transmettez la même configuration dans une dérogation d'application et dans les paramètres Hive, les paramètres Hive sont prioritaires. La liste suivante classe les configurations de la priorité la plus élevée à la plus faible.

- Configuration que vous fournissez dans le cadre des paramètres Hive avec`--hiveconf property=value`.
- La configuration que vous fournissez dans le cadre de votre application remplace lorsque vous démarrez une tâche.
- Configuration que vous fournissez `runtimeConfiguration` lors de la création d'une application.
- Configurations optimisées attribuées par Amazon EMR à la publication.
- Configurations open source par défaut pour l'application.

Pour plus d'informations sur la déclaration de configurations au niveau de l'application et sur le remplacement de configurations lors de l'exécution d'une tâche, reportez-vous àConfiguration d'application par défaut pour EMR Serverless.

Propriétés des tâches Hive

Le tableau suivant répertorie les propriétés obligatoires qui sont configurées lorsque vous soumettez une tâche Hive.

Propriétés obligatoires du job Hive

Paramètre	Description
`hive.exec.scratchdir`	Emplacement Amazon S3 où EMR Serverless crée des fichiers temporaires pendant l'exécuti on de la tâche Hive.
`hive.metastore.warehouse.dir`	Emplacement Amazon S3 des bases de données pour les tables gérées dans Hive.

Le tableau suivant répertorie les propriétés optionnelles de Hive et leurs valeurs par défaut que vous pouvez remplacer lorsque vous soumettez une tâche Hive.

Propriétés Hive facultatives et valeurs par défaut

Paramètre	Description	Valeur par défaut
`fs.s3.customAWSCre dentialsProvider`	Le fournisseur AWS d'informa tions d'identification que vous souhaitez utiliser.	com.amazonaws.auth.default AWSCredentials ProviderC hain
`fs.s3a.aws.credent ials.provider`	Le fournisseur AWS d'informa tions d'identification que vous souhaitez utiliser avec un système de fichiers S3A.	com.amazonaws.auth.default AWSCredentials ProviderC hain
`hive.auto.convert. join`	Option qui active la conversio n automatique des jointures communes en mapjoins, en fonction de la taille du fichier d'entrée.	`TRUE`
`hive.auto.convert. join.noconditional task`	Option qui active l'optimis ation lorsque Hive convertit une jointure commune en mapjoin en fonction de la taille du fichier d'entrée.	`TRUE`

Paramètre	Description	Valeur par défaut
`hive.auto.convert.join.noconditionaltask.size`	Une jointure est convertie directement en une jointure mapjoin inférieure à cette taille.	La valeur optimale est calculée en fonction de la mémoire des tâches Tez
`hive.cbo.enable`	Option qui active les optimisations basées sur les coûts avec le framework Calcite.	`TRUE`
`hive.cli.tez.session.async`	Possibilité de démarrer une session Tez en arrière-plan pendant que votre requête Hive se compile. Lorsqu'il est défini sur`false`, Tez AM démarre après la compilation de votre requête Hive.	`TRUE`
`hive.compute.query.using.stats`	Option qui active Hive pour répondre à certaines requêtes avec des statistiques stockées dans le métastore. Pour les statistiques de base, définissez `hive.stats.autogather` sur`TRUE`. Pour une collection de requêtes plus avancée, exécutez`analyze table queries`.	`TRUE`
`hive.default.fileformat`	Format de fichier par défaut pour les `CREATE TABLE` instructions. Vous pouvez le remplacer explicitement si vous le spécifiez `STORED AS [FORMAT]` dans votre `CREATE TABLE` commande.	`TEXTFILE`

Paramètre	Description	Valeur par défaut
`hive.driver.cores`	Nombre de cœurs à utiliser pour le processus du pilote Hive.	2
`hive.driver.disk`	Taille du disque pour le pilote Hive.	20G
`hive.driver.disk.type`	Type de disque pour le pilote Hive.	Standard
`hive.tez.disk.type`	La taille du disque pour les ouvriers.	Standard
`hive.driver.memory`	Quantité de mémoire à utiliser par processus de pilote Hive. La CLI Hive et le Tez Application Master partagent cette mémoire à parts égales avec 20 % de marge de manœuvre.	6 G
`hive.emr-serverless.launch.env.[ KEY]`	Option permettant de définir la variable d'*KEY*environnement dans tous les processus spécifiques à Hive, tels que votre pilote Hive, Tez AM et la tâche Tez.	
`hive.exec.dynamic.partition`	Options qui activent les partitions dynamiques en DML/DDL.	TRUE

Paramètre	Description	Valeur par défaut
`hive.exec.dynamic.partition.mode`	Option qui indique si vous souhaitez utiliser le mode strict ou le mode non strict. En mode strict, spécifiez au moins une partition statique au cas où vous remplaceriez accidentellement toutes les partitions. En mode non strict, toutes les partitions peuvent être dynamiques.	`strict`
`hive.exec.max.dynamic.partitions`	Le nombre maximum de partitions dynamiques créées par Hive au total.	1 000
`hive.exec.max.dynamic.partitions.pernode`	Nombre maximal de partitions dynamiques créées par Hive dans chaque nœud de mappage et de réduction.	100
`hive.exec.orc.split.strategy`	Attend l'une des valeurs suivantes : `BIETL`, ou`HYBRID`. Il ne s'agit pas d'une configuration au niveau de l'utilisateur. `BI`indique que vous souhaitez consacrer moins de temps à la génération fractionnée plutôt qu'à l'exécution de requêtes. `ETL`indique que vous souhaitez consacrer plus de temps à la génération fractionnée. `HYBRID`spécifie un choix des stratégies précédentes sur la base de l'heuristique.	`HYBRID`

Paramètre	Description	Valeur par défaut
`hive.exec.reducers.bytes.per.reducer`	La taille de chaque réducteur . La valeur par défaut est de 256 Mo. Si la taille d'entrée est de 1 G, la tâche utilise 4 réducteurs.	256000000
`hive.exec.reducers.max`	Le nombre maximum de réducteurs.	256
`hive.exec.stagingdir`	Le nom du répertoire qui stocke les fichiers temporaires créés par Hive dans les emplacements des tables et dans l'emplacement du répertoire de travail spécifié dans la `hive.exec.scratchdir` propriété.	`.hive-staging`
`hive.fetch.task.conversion`	Attend l'une des valeurs suivantes : `NONEMINIMAL`, ou`MORE`. Hive peut convertir certaines requêtes en une seule `FETCH` tâche. Cela permet de minimiser le temps de latence.	`MORE`
`hive.groupby.position.alias`	Option qui oblige Hive à utiliser un alias de position de colonne dans les `GROUP BY` instructions.	`FALSE`

Paramètre	Description	Valeur par défaut
`hive.input.format`	Format de saisie par défaut. Réglez sur `HiveInputFormat` si vous rencontrez des problèmes avec`CombineHiveInputFormat` .	`org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`
`hive.log.explain.output`	Option qui active les explications des sorties étendues pour toute requête de votre journal Hive.	`FALSE`
`hive.log.level`	Le niveau de journalisation de Hive.	`INFO`
`hive.mapred.reduce.tasks.speculative.execution`	Option qui active le lancement spéculatif pour les réducteurs. Compatible uniquement avec Amazon EMR 6.10.x et versions antérieures.	`TRUE`
`hive.max-task-containers`	Le nombre maximum de conteneurs simultanés. La mémoire du mappeur configurée est multipliée par cette valeur pour déterminer la mémoire disponible qui divise l'utilisation du calcul et de la préemption des tâches.	1 000
`hive.merge.mapfiles`	Option qui entraîne la fusion de petits fichiers à la fin d'une tâche cartographique uniquement.	`TRUE`

Paramètre	Description	Valeur par défaut
`hive.merge.size.per.task`	Taille des fichiers fusionnés à la fin de la tâche.	256000000
`hive.merge.tezfiles`	Option qui active la fusion de petits fichiers à la fin d'un Tez DAG.	`FALSE`
`hive.metastore.client.factory.class`	Nom de la classe d'usine qui produit les objets implémentant l'`IMetaStoreClient` interface.	`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`
`hive.metastore.glue.catalogid`	Si le catalogue de données AWS Glue agit comme un métastore mais s'exécute sous une autre forme Compte AWS que les tâches, l'ID de l' Compte AWS endroit où les tâches sont exécutées.	`NULL`
`hive.metastore.uris`	L'URI Thrift que le client du metastore utilise pour se connecter au metastore distant.	`NULL`
`hive.optimize.ppd`	Option qui active le pushdown des prédicats.	`TRUE`
`hive.optimize.ppd.storage`	Option qui active le transfert des prédicats vers les gestionnaires de stockage.	`TRUE`
`hive.orderby.position.alias`	Option qui oblige Hive à utiliser un alias de position de colonne dans les `ORDER BY` instructions.	`TRUE`

Paramètre	Description	Valeur par défaut
`hive.prewarm.enabled`	Option qui active le préchauffage du récipient pour Tez.	`FALSE`
`hive.prewarm.numcontainers`	Le nombre de récipients à préchauffer pour Tez.	10
`hive.stats.autogather`	Option permettant à Hive de collecter automatiquement des statistiques de base lors de la `INSERT OVERWRITE` commande.	`TRUE`
`hive.stats.fetch.column.stats`	Option qui désactive l'extraction des statistiques des colonnes depuis le métastore. L'extraction des statistiques des colonnes peut s'avérer coûteuse lorsque le nombre de colonnes est élevé.	`FALSE`
`hive.stats.gather.num.threads`	Nombre de threads utilisés par les commandes `partialscan` et `noscan` analyze pour les tables partitionnées. Cela ne s'applique qu'aux formats de fichiers implémentés `StatsProvidingRecordReader` (comme ORC).	10
`hive.strict.checks.cartesian.product`	Options qui activent des contrôles de jointure cartésiens stricts. Ces contrôles interdisent un produit cartésien (une jointure croisée).	`FALSE`

Paramètre	Description	Valeur par défaut
`hive.strict.checks.type.safety`	Option qui active des contrôles de sécurité de type stricts et désactive la comparaison entre `bigint` les deux `string` et`double`.	`TRUE`
`hive.support.quoted.identifiers`	Attend une valeur de `NONE` ou`COLUMN`. `NONE`implique que seuls les caractères alphanumériques et les traits de soulignement sont valides dans les identifiants. `COLUMN`implique que les noms de colonnes peuvent contenir n'importe quel caractère.	`COLUMN`
`hive.tez.auto.reducer.parallelism`	Option qui active la fonction de parallélisme du réducteur automatique Tez. Hive estime toujours la taille des données et définit des estimations de parallélisme. Tez échantillonne les tailles de sortie des sommets sources et ajuste les estimations au moment de l'exécution si nécessaire.	`TRUE`
`hive.tez.container.size`	La quantité de mémoire à utiliser par processus de tâche Tez.	6144
`hive.tez.cpu.vcores`	Le nombre de cœurs à utiliser pour chaque tâche Tez.	2
`hive.tez.disk.size`	Taille du disque pour chaque conteneur de tâches.	20G

Paramètre	Description	Valeur par défaut
`hive.tez.input.format`	Le format d'entrée pour la génération de splits dans le Tez AM.	`org.apache.hadoop.hive.ql.io.HiveInputFormat`
`hive.tez.min.partition.factor`	Limite inférieure de réducteurs spécifiée par Tez lorsque vous activez le parallélisme des réducteurs automatiques.	0.25
`hive.vectorized.execution.enabled`	Option qui active le mode vectorisé d'exécution des requêtes.	`TRUE`
`hive.vectorized.execution.reduce.enabled`	Option qui active le mode vectorisé du côté réduction de l'exécution d'une requête.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	Nom de classe de pilote pour un métastore JDBC.	`org.apache.derby.jdbc.EmbeddedDriver`
`javax.jdo.option.ConnectionPassword`	Le mot de passe associé à une base de données de métastore.	`NULL`
`javax.jdo.option.ConnectionURL`	Chaîne de connexion JDBC pour un métastore JDBC.	`jdbc:derby:;databaseName=metastore_db;create=true`
`javax.jdo.option.ConnectionUserName`	Nom d'utilisateur associé à une base de données Metastore.	`NULL`

Paramètre	Description	Valeur par défaut
`mapreduce.input.fileinputformat.split.maxsize`	La taille maximale d'une division lors du calcul de la division lorsque votre format d'entrée est `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`. La valeur 0 indique l'absence de limite.	0
`tez.am.dag.cleanup.on.completion`	Option qui active le nettoyage des données de shuffle lorsque le DAG est terminé.	`TRUE`
`tez.am.emr-serverless.launch.env.[KEY]`	Option permettant de définir la variable d'*KEY*environnement dans le processus Tez AM. Pour Tez AM, cette valeur remplace la `hive.emr-serverless.launch.env.[KEY]` valeur.	
`tez.am.log.level`	Le niveau de journalisation root qu'EMR Serverless transmet au serveur principal de l'application Tez.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMR Serverless devrait envoyer les événements ATS après cette période après la demande d'arrêt AM.	0

Paramètre	Description	Valeur par défaut
`tez.am.speculation.enabled`	Option qui entraîne le lancement spéculatif de tâches plus lentes. Cela peut aider à réduire la latence des tâches lorsque certaines tâches s'exécutent plus lentement en raison de machines défectueuses ou lentes. Compatible uniquement avec Amazon EMR 6.10.x et versions antérieures.	`FALSE`
`tez.am.task.max.failed.attempts`	Nombre maximal de tentatives susceptibles d'échouer pour une tâche donnée avant que celle-ci n'échoue. Ce nombre ne compte pas les tentatives annulées manuellement.	3
`tez.am.vertex.cleanup.height`	Distance à laquelle, si tous les sommets dépendants sont complets, Tez AM supprimera les données de mélange de sommets. Cette fonctionnalité est désactivée lorsque la valeur est 0. Les versions 6.8.0 et ultérieures d'Amazon EMR prennent en charge cette fonctionnalité.	0
`tez.client.asynchronous-stop`	Option permettant à EMR Serverless de transmettre les événements ATS avant de mettre fin au pilote Hive.	`FALSE`

Paramètre	Description	Valeur par défaut
`tez.grouping.max-size`	Limite de taille supérieure (en octets) d'un fractionnement groupé. Cette limite permet d'éviter des divisions trop importantes.	1073741824
`tez.grouping.min-size`	Limite de taille inférieure (en octets) d'un fractionnement groupé. Cette limite permet d'éviter un trop grand nombre de petites divisions.	16777216
`tez.runtime.io.sort.mb`	La taille de la mémoire tampon souple lorsque Tez trie la sortie est triée.	La valeur optimale est calculée sur la base de la mémoire des tâches Tez
`tez.runtime.unordered.output.buffer.size-mb`	La taille de la mémoire tampon à utiliser si Tez n'écrit pas directement sur le disque.	La valeur optimale est calculée sur la base de la mémoire des tâches Tez

Paramètre	Description	Valeur par défaut
`tez.shuffle-vertex-manager.max-src-fraction`	Fraction des tâches source qui doivent être effectuées avant qu'EMR Serverless ne planifie toutes les tâches pour le sommet actuel (dans le cas d'une connexion). `ScatterGather` Le nombre de tâches prêtes à être planifiées sur le sommet actuel varie linéairement entre et`min-fraction` . `max-fraction` Cela correspond à la valeur par défaut ou `tez.shuffle-vertex-manager.min-src-fraction` à la valeur la plus élevée.	0.75
`tez.shuffle-vertex-manager.min-src-fraction`	Fraction des tâches source qui doivent être effectuées avant qu'EMR Serverless ne planifie des tâches pour le sommet actuel (dans le cas d'une connexion). `ScatterGather`	0.25
`tez.task.emr-serverless.launch.env.[` *KEY*`]`	Option permettant de définir la variable d'*KEY*environnement dans le processus de tâche Tez. Pour les tâches Tez, cette valeur remplace la `hive.emr-serverless.launch.env.[` *KEY*`]` valeur.	

Paramètre	Description	Valeur par défaut
`tez.task.log.level`	Le niveau de journalisation racine qu'EMR Serverless transmet aux tâches Tez.	`INFO`
`tez.yarn.ats.event.flush.timeout.millis`	Durée maximale pendant laquelle AM doit attendre que les événements soient supprimés avant de s'arrêter.	300 000

Exemples d'emplois chez Hive

L'exemple de code suivant montre comment exécuter une requête Hive avec l'`StartJobRun`API.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-
query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/
hive/scratch",
                "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1"
            }
        }]
    }'
```

Vous trouverez d'autres exemples d'exécution de tâches Hive dans le référentiel EMR Serverless GitHub Samples.

Résilience des tâches EMR sans serveur

Les versions 7.1.0 et supérieures d'EMR Serverless incluent la prise en charge de la résilience des tâches, de sorte qu'il réessaie automatiquement toutes les tâches ayant échoué sans aucune intervention manuelle de votre part. Un autre avantage de la résilience des tâches est qu'EMR Serverless déplace les exécutions de tâches vers une autre zone de disponibilité (AZ) en cas de problème avec une AZ.

Pour activer la résilience d'une tâche, définissez la politique de nouvelle tentative pour cette tâche. Une politique de nouvelles tentatives garantit qu'EMR Serverless redémarre automatiquement une tâche en cas d'échec. Les politiques de nouvelle tentative sont prises en charge pour les tâches par lots et en streaming. Personnalisez donc la résilience des tâches en fonction de votre cas d'utilisation. Le tableau suivant compare les comportements et les différences de résilience des tâches entre les tâches par lots et les tâches de streaming.

	Traitements par lots	Offres d'emploi en streaming
Comportement par défaut	Ne réexécute pas le job.	Réessaie toujours d'exécuter la tâche car l'application crée des points de contrôle pendant l'exécution de la tâche.
Point de nouvelle tentative	Les tâches par lots ne comportant pas de points de contrôle, EMR Serverless réexécute toujours la tâche depuis le début.	Les tâches de streaming prennent en charge les points de contrôle. Configurez donc la requête de streaming pour enregistrer l'état d'exécution et progresser vers un point de contrôle dans Amazon S3. EMR Serverless reprend l'exécution du job depuis le point de contrôle. Pour plus d'informations, reportez-vous à la section Restaurer après un échec avec le point de

	Traitements par lots	Offres d'emploi en streaming
		contrôle dans la documenta tion d'Apache Spark.
Nombre maximum de tentative s de nouvelle tentative	Permet un maximum de 10 tentatives.	Les tâches de streaming sont dotées d'un contrôle intégré de prévention des thrash, de sorte que l'application arrête de réessayer les tâches si elles échouent toujours au bout d'une heure. Le nombre de tentatives par défaut dans un délai d'une heure est de cinq tentatives. Vous pouvez configurer ce nombre de tentatives pour qu'il soit compris entre 1 et 10. Vous ne pouvez pas personnaliser le nombre maximum de tentative s. La valeur 1 indique qu'il n'y a pas eu de nouvelle tentative.

Lorsque EMR Serverless tente de réexécuter une tâche, il indexe également la tâche à l'aide d'un numéro de tentative. Suivez donc le cycle de vie d'une tâche au fil de ses tentatives.

Utilisez les opérations de l'API EMR Serverless ou AWS CLI pour modifier la résilience des tâches ou accéder aux informations relatives à la résilience des tâches. Pour plus d'informations, consultez le guide de l'API EMR Serverless.

Par défaut, EMR Serverless ne réexécute pas les tâches par lots. Pour activer les nouvelles tentatives pour les tâches par lots, configurez le `maxAttempts` paramètre lorsque vous démarrez l'exécution d'une tâche par lots. Le `maxAttempts` paramètre s'applique uniquement aux tâches par lots. La valeur par défaut est 1, ce qui signifie qu'il ne faut pas réexécuter la tâche. Les valeurs acceptées sont comprises entre 1 et 10 inclus.

L'exemple suivant montre comment spécifier un nombre maximum de 10 tentatives lors du démarrage d'une tâche.

```
aws emr-serverless start-job-run
 --application-id <APPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'BATCH' \
 --retry-policy '{
    "maxAttempts": 10
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

EMR Serverless réessaie indéfiniment les tâches de streaming en cas d'échec. Pour éviter le thrash dû à des échecs irrécupérables répétés, utilisez le `maxFailedAttemptsPerHour` pour configurer le contrôle de prévention du thrash pour les nouvelles tentatives de travail en streaming. Ce paramètre vous permet de spécifier le nombre maximum de tentatives infructueuses autorisées une heure avant qu'EMR Serverless arrête de réessayer. La valeur par défaut est cinq. Les valeurs acceptées sont comprises entre 1 et 10 inclus.

```
aws emr-serverless start-job-run
 --application-id <APPPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'STREAMING' \
 --retry-policy '{
    "maxFailedAttemptsPerHour": 7
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

Vous pouvez également utiliser les autres opérations d'API d'exécution de tâches pour obtenir des informations sur les tâches. Par exemple, utilisez le `attempt` paramètre avec l'`GetJobRun`opération

pour obtenir des informations sur une tentative de travail spécifique. Si vous n'incluez pas le `attempt` paramètre, l'opération renvoie des informations sur la dernière tentative.

```
aws emr-serverless get-job-run \
    --job-run-id job-run-id \
    --application-id application-id \
    --attempt 1
```

L'`ListJobRunAttempts`opération renvoie des informations sur toutes les tentatives liées à l'exécution d'une tâche.

```
aws emr-serverless list-job-run-attempts \
  --application-id application-id \
  --job-run-id job-run-id
```

L'`GetDashboardForJobRun`opération crée et renvoie une URL qui permet d'accéder à l'application UIs pour l'exécution d'une tâche. Le `attempt` paramètre vous permet d'obtenir l'URL d'une tentative spécifique. Si vous n'incluez pas le `attempt` paramètre, l'opération renvoie des informations sur la dernière tentative.

```
aws emr-serverless get-dashboard-for-job-run \
    --application-id application-id \
    --job-run-id job-run-id \
    --attempt 1
```

Surveillance d'une tâche à l'aide d'une politique de relance

Le support de résilience des tâches ajoute également le nouvel événement EMR Serverless job run retry. EMR Serverless publie cet événement à chaque nouvelle tentative de la tâche. Vous pouvez utiliser cette notification pour suivre les nouvelles tentatives de la tâche. Pour plus d'informations sur les événements, consultez la section Amazon EventBridge events.

Connexion avec politique de nouvelle tentative

Chaque fois qu'EMR Serverless réessaie une tâche, la tentative génère son propre ensemble de journaux. Pour garantir qu'EMR Serverless puisse transmettre ces journaux à Amazon S3 et Amazon CloudWatch sans en remplacer aucun, EMR Serverless ajoute un préfixe au format du chemin du journal S3 et au nom du flux de journal pour inclure le numéro de CloudWatch tentative de la tâche.

Voici un exemple de ce à quoi ressemble le format.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

Ce format garantit qu'EMR Serverless publie tous les journaux de chaque tentative de tâche vers son propre emplacement désigné dans Amazon S3 et. CloudWatch Pour plus de détails, reportez-vous à la section Stockage des journaux.

> **Note**
>
> EMR Serverless utilise uniquement ce format de préfixe pour toutes les tâches de streaming et les tâches par lots pour lesquelles la nouvelle tentative est activée.

Configuration du métastore pour EMR Serverless

Un métastore Hive est un emplacement centralisé qui stocke les informations structurelles relatives à vos tables, notamment les schémas, les noms de partitions et les types de données. Avec EMR Serverless, vous pouvez conserver les métadonnées de cette table dans un métastore ayant accès à vos tâches.

Deux options s'offrent à vous pour créer un métastore Hive :

- Le catalogue AWS de données Glue
- Un métastore Apache Hive externe

Utiliser le catalogue AWS de données Glue comme métastore

Vous pouvez configurer vos tâches Spark et Hive pour utiliser le catalogue de données AWS Glue comme métastore. Nous recommandons cette configuration lorsque vous avez besoin d'un métastore persistant ou d'un métastore partagé par différentes applications, services ou. Comptes AWS Pour plus d'informations sur le catalogue de données, reportez-vous à la section Remplissage du catalogue de données AWS Glue. Pour plus d'informations sur AWS la tarification de Glue, reportez-vous à AWS la section Tarification de Glue.

Vous pouvez configurer votre tâche EMR Serverless pour utiliser le catalogue de données AWS Glue de la même manière Compte AWS que votre application ou d'une autre manière. Compte AWS

Configuration du catalogue AWS de données Glue

Pour configurer le catalogue de données, choisissez le type d'application EMR sans serveur que vous souhaitez utiliser.

Spark

Lorsque vous utilisez EMR Studio pour exécuter vos tâches avec des applications EMR Serverless Spark, le catalogue de données AWS Glue est le métastore par défaut.

Lorsque vous utilisez SDKs ou AWS CLI, définissez la `spark.hadoop.hive.metastore.client.factory.class` configuration sur `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` dans les `sparkSubmit` paramètres de votre tâche d'exécution. L'exemple suivant montre comment configurer le catalogue de données avec AWS CLI.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/code/pyspark/
extreme_weather.py",
            "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
 --conf spark.driver.cores=1 --conf spark.driver.memory=3g --conf
 spark.executor.cores=4 --conf spark.executor.memory=3g"
        }
    }'
```

Vous pouvez également définir cette configuration lorsque vous créez une nouvelle configuration `SparkSession` dans votre code Spark.

```
from pyspark.sql import SparkSession

spark = (
    SparkSession.builder.appName("SparkSQL")
    .config(
        "spark.hadoop.hive.metastore.client.factory.class",
        "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
    )
    .enableHiveSupport()
```

```
    .getOrCreate()
)

# we can query tables with SparkSQL
spark.sql("SHOW TABLES").show()

# we can also them with native Spark
print(spark.catalog.listTables())
```

Hive

Pour les applications EMR Serverless Hive, le catalogue de données est le métastore par défaut. En d'autres termes, lorsque vous exécutez des tâches sur une application EMR Serverless Hive, Hive enregistre les informations du métastore dans le catalogue de données au même titre que votre application. Compte AWS Vous n'avez pas besoin d'un cloud privé virtuel (VPC) pour utiliser le catalogue de données comme métastore.

Pour accéder aux tables du métastore Hive, ajoutez les politiques AWS Glue requises décrites dans Configuration des autorisations IAM pour Glue. AWS

Configuration de l'accès entre comptes pour EMR Serverless AWS et Glue Data Catalog

Pour configurer l'accès entre comptes pour EMR Serverless, connectez-vous d'abord à l'adresse suivante : Comptes AWS

- `AccountA`— Et Compte AWS où vous avez créé une application EMR Serverless.
- `AccountB`— Et Compte AWS qui contient un catalogue de données AWS Glue auquel vous souhaitez que votre tâche EMR Serverless puisse accéder.

1. Assurez-vous qu'un administrateur ou une autre identité autorisée `AccountB` attache une politique de ressources au catalogue de données dans`AccountB`. Cette politique accorde `AccountA` des autorisations spécifiques entre comptes pour effectuer des opérations sur les ressources du `AccountB` catalogue.

   JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "arn:aws:glue:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

2. Ajoutez une politique IAM au rôle d'exécution des tâches EMR sans serveur afin que ce rôle puisse accéder aux ressources du catalogue de données `AccountA` dans. `AccountB`

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
```

```
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "arn:aws:glue:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

3. Commencez à exécuter votre tâche. Cette étape est légèrement différente selon le `AccountA` type d'application EMR Serverless de l'application.

   Spark

   Transmettez la `spark.hadoop.hive.metastore.glue.catalogid` propriété `sparkSubmitParameters` comme indiqué dans l'exemple suivant. Remplacez *`AccountB-catalog-id`* par l'ID du catalogue de données dans`AccountB`.

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://amzn-s3-demo-bucket/scripts/test.py",
         "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.A
 --conf spark.hadoop.hive.metastore.glue.catalogid=AccountB-catalog-
id --conf spark.executor.cores=1 --conf spark.executor.memory=1g
 --conf spark.driver.cores=1 --conf spark.driver.memory=1g --conf
 spark.executor.instances=1"
    }
  }' \
--configuration-overrides '{
    "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
```

```
    "logUri": "s3://amzn-s3-demo-bucket/logs/"
    }
  }
}'
```

Hive

Définissez la `hive.metastore.glue.catalogid` propriété dans la `hive-site` classification comme indiqué dans l'exemple suivant. Remplacez *AccountB-catalog-id* par l'ID du catalogue de données dans`AccountB`.

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "hive": {
    "query": "s3://amzn-s3-demo-bucket/hive/scripts/create_table.sql",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
hive/warehouse"
    }
}' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

Considérations relatives à l'utilisation du catalogue AWS de données Glue

Vous pouvez ajouter JARs un auxiliaire `ADD JAR` dans vos scripts Hive. Pour plus d'informations, reportez-vous à la section Considérations relatives à l'utilisation de AWS Glue Data Catalog.

Utilisation d'un métastore Hive externe

Vous pouvez configurer vos tâches EMR Serverless Spark et Hive pour vous connecter à un métastore Hive externe, tel qu'Amazon Aurora ou Amazon RDS for MySQL. Cette section explique

comment configurer un métastore Amazon RDS Hive, configurer votre VPC et configurer vos tâches EMR Serverless pour utiliser un métastore externe.

Création d'un métastore Hive externe

1. Créez un Amazon Virtual Private Cloud (Amazon VPC) avec des sous-réseaux privés en suivant les instructions de la section Créer un VPC.
2. Créez votre application EMR Serverless avec votre nouvel Amazon VPC et vos sous-réseaux privés. Lorsque vous configurez votre application EMR Serverless avec un VPC, elle fournit d'abord une interface Elastic Network pour chaque sous-réseau que vous spécifiez. Il attache ensuite le groupe de sécurité que vous avez spécifié à cette interface réseau. Cela permet à votre application de contrôler l'accès. Pour plus de détails sur la configuration de votre VPC, reportez-vous à. Configuration de l'accès VPC pour que les applications EMR sans serveur se connectent aux données
3. Créez une base de données MySQL ou Aurora PostgreSQL dans un sous-réseau privé de votre Amazon VPC. Pour plus d'informations sur la création d'une base de données Amazon RDS, reportez-vous à la section Création d'une instance de base de données Amazon RDS.
4. Modifiez le groupe de sécurité de votre base de données MySQL ou Aurora pour autoriser les connexions JDBC depuis votre groupe de sécurité EMR Serverless en suivant les étapes décrites dans Modification d'une instance de base de données Amazon RDS. Ajoutez une règle pour le trafic entrant au groupe de sécurité RDS depuis l'un de vos groupes de sécurité EMR Serverless.

Type	Protocole	Plage de ports	Source
Tous les TCP	TCP	3306	`emr-serverless-security-group`

Configuration des options Spark

Utilisation de JDBC

Pour configurer votre application EMR Serverless Spark afin de se connecter à un metastore Hive basé sur une instance Amazon RDS for MySQL ou Amazon Aurora MySQL, utilisez une connexion

JDBC. Transmettez le `mariadb-connector-java.jar` with `--jars` dans les `spark-submit` paramètres de votre exécution de tâche.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-
connector-java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=<connection-user-
name>
            --conf spark.hadoop.javax.jdo.option.ConnectionPassword=<connection-
password>
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=<JDBC-Connection-
string>
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

L'exemple de code suivant est un script Spark entrypoint qui interagit avec un métastore Hive sur Amazon RDS.

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
```

```
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE `sampledb`.`sparknyctaxi`(`dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT count(*) FROM sampledb.sparknyctaxi").show()
spark.stop()
```

Utilisation du service d'épargne

Vous pouvez configurer votre application EMR Serverless Hive pour qu'elle se connecte à un métastore Hive basé sur une instance Amazon RDS for MySQL ou Amazon Aurora MySQL. Pour ce faire, exécutez un serveur d'épargne sur le nœud principal d'un cluster Amazon EMR existant. Cette option est idéale si vous possédez déjà un cluster Amazon EMR avec un serveur d'épargne que vous souhaitez utiliser pour simplifier vos configurations de tâches EMR sans serveur.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/thriftscript.py",
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-
connector-java.jar
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

L'exemple de code suivant est un script entrypoint (`thriftscript.py`) qui utilise le protocole Thrift pour se connecter à un métastore Hive. Notez que la `hive.metastore.uris` propriété doit être définie pour être lue depuis un métastore Hive externe.

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris","thrift://thrift-server-host:thift-server-port") \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE sampledb.`sparknyctaxi`( `dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT * FROM sampledb.sparknyctaxi").show()
spark.stop()
```

Configurer les options Hive

Utilisation de JDBC

Si vous souhaitez spécifier un emplacement de base de données Hive externe sur une instance Amazon RDS MySQL ou Amazon Aurora, vous pouvez remplacer la configuration de métastore par défaut.

> **Note**
>
> Dans Hive, vous pouvez effectuer plusieurs écritures dans des tables de métastore en même temps. Si vous partagez des informations de métastore entre deux jobs, assurez-vous de ne pas écrire simultanément dans la même table de métastore, sauf si vous écrivez sur des partitions différentes de la même table de métastore.

Définissez les configurations suivantes dans la `hive-site` classification pour activer le métastore Hive externe.

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "password"
    }
}
```

Utilisation d'un serveur d'épargne

Vous pouvez configurer votre application EMR Serverless Hive pour qu'elle se connecte à un métastore Hive basé sur Amazon RDS for MySQL ou Amazon Aurora My. SQLinstance Pour ce faire, exécutez un serveur d'épargne sur le nœud principal d'un cluster Amazon EMR existant. Cette option est idéale si vous possédez déjà un cluster Amazon EMR qui exécute un serveur d'épargne et que vous souhaitez utiliser vos configurations de tâches EMR sans serveur.

Définissez les configurations suivantes dans la `hive-site` classification afin qu'EMR Serverless puisse accéder à la métastore d'épargne distante. Notez que vous devez définir la `hive.metastore.uris` propriété pour qu'elle puisse être lue depuis un métastore Hive externe.

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "hive.metastore.uris": "thrift://thrift-server-host:thirft-server-port"
    }
}
```

Utilisation de la hiérarchie multi-catalogues AWS Glue sur EMR Serverless

Vous pouvez configurer vos applications EMR Serverless pour qu'elles fonctionnent avec la hiérarchie multi-catalogues AWS Glue. L'exemple suivant montre comment utiliser EMR-S Spark avec la hiérarchie multi-catalogues AWS Glue.

Pour en savoir plus sur la hiérarchie de plusieurs catalogues, reportez-vous à la section Utilisation d'une hiérarchie de catalogues multiples dans AWS Glue Data Catalog with Spark sur Amazon EMR.

Utilisation du stockage géré Redshift (RMS) avec Iceberg et AWS Glue Data Catalog

Voici comment configurer Spark pour l'intégrer à un catalogue de données AWS Glue avec Iceberg :

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/myscript.py",
            "sparkSubmitParameters": "--conf spark.sql.catalog.nfgac_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.defaultCatalog=rms
             --conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
        }
    }'
```

Exemple de requête provenant d'une table du catalogue, suite à l'intégration :

```
SELECT * FROM my_rms_schema.my_table
```

Utilisation du stockage géré Redshift (RMS) avec l'API REST d'Iceberg et le catalogue de données Glue AWS

Voici comment configurer Spark pour qu'il fonctionne avec le catalogue REST d'Iceberg :

```
aws emr-serverless start-job-run \
--application-id application-id \
--execution-role-arn job-role-arn \
--job-driver '{
"sparkSubmit": {
"entryPoint": "s3://amzn-s3-demo-bucket/myscript.py",
    "sparkSubmitParameters": "
    --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"
    }
 }'
```

Exemple de requête provenant d'une table du catalogue :

```
SELECT * FROM my_rms_schema.my_table
```

Considérations relatives à l'utilisation d'un métastore externe

- Vous pouvez configurer des bases de données compatibles avec MariaDB JDBC en tant que metastore. RDS pour MariaDB, MySQL et Amazon Aurora sont des exemples de ces bases de données.
- Les métastores ne sont pas initialisés automatiquement. Si votre métastore n'est pas initialisé avec un schéma pour votre version de Hive, utilisez l'outil Hive Schema.
- EMR Serverless ne prend pas en charge l'authentification Kerberos. Vous ne pouvez pas utiliser un serveur Thrift Metastore avec authentification Kerberos avec des tâches EMR Serverless Spark ou Hive.
- Vous devez configurer l'accès au VPC pour utiliser la hiérarchie multi-catalogues.

Accès aux données S3 depuis un autre AWS compte depuis EMR Serverless

Vous pouvez exécuter des tâches Amazon EMR sans serveur à partir d'un AWS compte et les configurer pour accéder aux données des compartiments Amazon S3 appartenant à un autre compte. AWS Cette page décrit comment configurer l'accès entre comptes à S3 depuis EMR Serverless.

Les tâches exécutées sur EMR Serverless peuvent utiliser une politique de compartiment S3 ou un rôle assumé pour accéder aux données dans Amazon S3 à partir d'un autre compte. AWS

Conditions préalables

Pour configurer l'accès entre comptes pour Amazon EMR Serverless, effectuez les tâches en étant connecté à deux comptes : AWS

- **`AccountA`**— Il s'agit du AWS compte sur lequel vous avez créé une application Amazon EMR Serverless. Avant de configurer l'accès entre comptes, préparez les éléments suivants dans ce compte :
  - Une application Amazon EMR Serverless dans laquelle vous souhaitez exécuter des tâches.
  - Rôle d'exécution de tâches disposant des autorisations requises pour exécuter des tâches dans l'application. Pour plus d'informations, consultez Rôles d'exécution des tâches pour Amazon EMR Serverless.
- **`AccountB`**— Il s'agit du AWS compte qui contient le compartiment S3 auquel vous souhaitez que vos tâches Amazon EMR Serverless accèdent.

Utiliser une politique de compartiment S3 pour accéder aux données S3 entre comptes

Pour accéder au compartiment S3 account B depuisaccount A, attachez la politique suivante au compartiment S3 depuisaccount B.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePermissions1",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-name"
      ]
    },
    {
      "Sid": "ExamplePermissions2",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
```

```
      "Resource": [
        "arn:aws:s3:::my-bucket-name/*"
      ]
    }
  ]
}
```

Pour plus d'informations sur l'accès entre comptes S3 avec les politiques relatives aux compartiments S3, reportez-vous à l'exemple 2 : le propriétaire du compartiment accorde des autorisations de compartiment entre comptes dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Utiliser un rôle assumé pour accéder aux données S3 entre comptes

Une autre méthode pour configurer l'accès entre comptes pour Amazon EMR Serverless consiste à utiliser `AssumeRole` l'action du AWS Security Token Service ().AWS STS AWS STS est un service Web mondial qui vous permet de demander des informations d'identification temporaires à privilèges limités pour les utilisateurs. Vous pouvez effectuer des appels d'API vers EMR Serverless et Amazon S3 à l'aide des informations d'identification de sécurité temporaires que vous avez créées avec. `AssumeRole`

Les étapes suivantes illustrent comment utiliser un rôle assumé pour accéder aux données S3 entre comptes depuis EMR Serverless :

1. Créez un compartiment Amazon S3, *`cross-account-bucket`*, dans `AccountB`. Pour plus d'informations, reportez-vous à la section Création d'un compartiment dans le guide de l'utilisateur d'Amazon Simple Storage Service. Si vous souhaitez bénéficier d'un accès multicompte à DynamoDB, créez également une table DynamoDB dans. `AccountB` Pour plus d'informations, reportez-vous à la section Création d'une table DynamoDB dans le manuel du développeur Amazon DynamoDB.
2. Créez un rôle IAM `Cross-Account-Role-B` dans le `AccountB` qui peut accéder au *`cross-account-bucket`*.
   a. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
   b. Choisissez Rôles et créez un nouveau rôle : `Cross-Account-Role-B`. Pour plus d'informations sur la création de rôles IAM, reportez-vous à la section Création de rôles IAM dans le Guide de l'utilisateur IAM.

c. 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, reportez-vous à la section Création de politiques IAM 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 vous avez besoin d'un accès DynamoDB, créez une politique IAM qui spécifie les autorisations d'accès à la table DynamoDB entre comptes. Attachez ensuite la politique IAM au `Cross-Account-Role-B`. Pour plus d'informations, consultez Amazon DynamoDB : autorise l'accès à une table spécifique dans le guide de l'utilisateur IAM.

Voici une politique permettant d'autoriser l'accès à la table DynamoDB`CrossAccountTable`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:*"
      ],
```

```
      "Resource": [
        "arn:aws:dynamodb:*:123456789012:table/CrossAccountTable"
      ],
      "Sid": "AllowDYNAMODB"
    }
  ]
}
```

3. Modifiez la relation de confiance du rôle `Cross-Account-Role-B`.

   a. Pour configurer la relation de confiance pour le rôle, choisissez l'onglet Relations de confiance dans la console IAM pour le rôle `Cross-Account-Role-B` que vous avez créé à l'étape 2.

   b. Sélectionnez Modifier la relation de confiance.

   c. Ajoutez le document de politique suivant. Cela permet `AccountA` à `Job-Execution-Role-A` in d'assumer le `Cross-Account-Role-B` rôle.

      JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/Job-Execution-Role-A",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

4. Accordez `Job-Execution-Role-A` `AccountA` l' AWS STS `AssumeRole`autorisation d'assumer`Cross-Account-Role-B`.

   a. Dans la console IAM du AWS compte`AccountA`, sélectionnez`Job-Execution-Role-A`.

   b. 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::123456789012:role/Cross-Account-Role-B"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

Exemples de rôles supposés

Utilisez un rôle assumé unique pour accéder à toutes les ressources S3 d'un compte, ou avec Amazon EMR 6.11 et versions ultérieures, configurez plusieurs rôles IAM à assumer lorsque vous accédez à différents compartiments S3 entre comptes.

Rubriques

- Accédez aux ressources S3 avec un rôle assumé
- Accédez aux ressources S3 avec plusieurs rôles assumés

Accédez aux ressources S3 avec un rôle assumé

> **Note**
>
> Lorsque vous configurez une tâche pour utiliser un seul rôle assumé, toutes les ressources S3 de la tâche utilisent ce rôle, y compris le `entryPoint` script.

Si vous souhaitez utiliser un seul rôle assumé pour accéder à toutes les ressources S3 du compte B, spécifiez les configurations suivantes :

1. Spécifiez la configuration `fs.s3.customAWSCredentialsProvider` EMRFS à. `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider`

2. Pour Spark, utilisez `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` et `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` pour spécifier les variables d'environnement sur le pilote et les exécuteurs.
3. Pour Hive, utilisez `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, et pour spécifier les variables `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` d'environnement sur le pilote Hive, l'application principale Tez et les conteneurs de tâches Tez.

Les exemples suivants montrent comment utiliser un rôle assumé pour démarrer une tâche EMR sans serveur exécutée avec un accès entre comptes.

Spark

L'exemple suivant montre comment utiliser un rôle assumé pour démarrer une tâche EMR Serverless Spark exécutée avec un accès multicompte à S3.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "entrypoint_location",
            "entryPointArguments": [":argument_1:", ":argument_2:"],
            "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
 --conf spark.executor.instances=1"
        }
    }' \
     --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "spark-defaults",
            "properties": {
                "spark.hadoop.fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.AssumeRoleAWSCredentialsProvider",
                "spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
```

```
                "spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Hive

L'exemple suivant montre comment utiliser un rôle assumé pour démarrer une tâche EMR Serverless Hive exécutée avec un accès multicompte à S3.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.task.emr-
serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Accédez aux ressources S3 avec plusieurs rôles assumés

Avec les versions 6.11.0 et supérieures d'EMR Serverless, configurez plusieurs rôles IAM à assumer lorsque vous accédez à différents buckets multicomptes. Si vous souhaitez accéder à différentes

ressources S3 avec différents rôles assumés dans le compte B, utilisez les configurations suivantes lorsque vous démarrez l'exécution de la tâche :

1. Spécifiez la configuration `fs.s3.customAWSCredentialsProvider` EMRFS à. `com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredenti`
2. Spécifiez la configuration EMRFS `fs.s3.bucketLevelAssumeRoleMapping` pour définir le mappage entre le nom du compartiment S3 et le rôle IAM à assumer dans le compte B. La valeur doit être au format de`bucket1->role1;bucket2->role2`.

Par exemple, utilisez `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` pour accéder au bucket `bucket1` et utilisez `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` pour accéder au bucket`bucket2`. Les exemples suivants montrent comment démarrer une tâche EMR sans serveur exécutée avec un accès entre comptes via plusieurs rôles assumés.

Spark

L'exemple suivant montre comment utiliser plusieurs rôles assumés pour créer une tâche EMR Serverless Spark exécutée.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "entrypoint_location",
            "entryPointArguments": [":argument_1:", ":argument_2:"],
            "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
 --conf spark.executor.instances=1"
        }
    }' \
     --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "spark-defaults",
            "properties": {
                "spark.hadoop.fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredentialsProvider"
                "spark.hadoop.fs.s3.bucketLevelAssumeRoleMapping":
 "bucket1->arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
```

```
        }]
    }'
```

Hive

Les exemples suivants montrent comment utiliser plusieurs rôles assumés pour créer une tâche EMR Serverless Hive exécutée.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "fs.s3.bucketLevelAssumeRoleMapping": "bucket1-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
        }]
    }'
```

Résolution des erreurs dans EMR Serverless

Utilisez les informations suivantes pour diagnostiquer et résoudre les problèmes courants qui se produisent lors de l'utilisation d'Amazon EMR Serverless.

Rubriques

- Erreur : Le travail a échoué car le compte a atteint la limite de service du nombre maximal de vCPU qu'il peut utiliser simultanément.
- Erreur : Le Job a échoué car l'application a dépassé les paramètres de capacité maximale.

Erreur : Le travail a échoué car le compte a atteint la limite de service du nombre maximal de vCPU qu'il peut utiliser simultanément.

Cette erreur indique qu'EMR Serverless n'a pas pu soumettre la tâche car le compte a dépassé la capacité maximale. Augmentez la capacité maximale du compte. Vérifiez vos limites de service en fonction des quotas de service EMR Serverless.

Erreur : Le Job a échoué car l'application a dépassé les paramètres de capacité maximale.

Cette erreur indique qu'EMR Serverless n'a pas pu soumettre la tâche car l'application a dépassé la capacité maximale configurée. Augmentez la capacité maximale de l'application.

Erreur : La tâche a échoué car le travailleur n'a pas pu être alloué car l'application a dépassé sa capacité maximale.

Cette erreur indique que le travail n'a pas pu être terminé. Les travailleurs n'ont pas pu être alloués car l'application a dépassé les paramètres de capacité maximale.

Erreur : l'accès S3 est refusé. Vérifiez les autorisations d'accès S3 du rôle d'exécution des tâches sur les ressources S3 requises.

Cette erreur indique que votre tâche n'a pas accès à vos ressources S3. Vérifiez que le rôle d'exécution de la tâche est autorisé à accéder aux ressources S3 que la tâche doit utiliser. Pour en savoir plus sur les rôles d'exécution, reportez-vous àRôles d'exécution des tâches pour Amazon EMR Serverless.

Erreur ModuleNotFoundError : aucun module n'a été nommé<module>. Reportez-vous au guide de l'utilisateur pour savoir comment utiliser les bibliothèques Python avec EMR Serverless.

Cette erreur indique qu'aucun module Python n'était disponible pour le job Spark. Vérifiez que les bibliothèques Python dépendantes sont disponibles pour la tâche. Pour plus d'informations sur la façon d'empaqueter des bibliothèques Python, reportez-vous àUtilisation de bibliothèques Python avec EMR Serverless.

Erreur : Impossible d'assumer le rôle d'exécution <role name>car celui-ci n'existe pas ou n'est pas configuré avec la relation de confiance requise.

Cette erreur indique que le rôle d'exécution de la tâche que vous avez spécifié pour la tâche n'existe pas ou qu'il n'a aucune relation de confiance pour les autorisations EMR Serverless. Pour vérifier que le rôle IAM existe et que vous avez correctement configuré la politique de confiance du rôle, consultez les instructions dansRôles d'exécution des tâches pour Amazon EMR Serverless.

Permettre la répartition des coûts au niveau des tâches

La répartition des coûts au niveau des tâches permet d'attribuer la facturation granulaire pour EMR Serverless au niveau de l'exécution individuelle des tâches, plutôt que d'agréger tous les coûts au niveau de l'application. Lorsque cette option est activée, vous pouvez filtrer et suivre les AWS coûts dans Cost Explorer et Cost and Usage Reports en fonction de l'exécution des tâches spécifiques IDs et des balises associées aux séries de tâches, offrant ainsi une meilleure visibilité des frais liés aux séries de tâches soumises.

Comportement par défaut

La répartition des coûts au niveau des tâches n'est pas activée par défaut.

Comment activer ou désactiver la fonctionnalité

Vous pouvez configurer la répartition des coûts au niveau des tâches lors de la création de l'application ou la mettre à jour pour les applications existantes.

Spécifiez le `jobLevelCostAllocation` paramètre lors de la création d'une nouvelle application :

```
# Enable job-level cost allocation:
```

```
aws emr-serverless create-application \
    --name "my-application" \
    --release-label "emr-7.12.0" \
    --type "SPARK" \
    --job-level-cost-allocation-configuration '{
        "enabled": true
    }'

# Disable job-level cost allocation:
aws emr-serverless create-application \
    --name "my-application" \
    --release-label "emr-7.12.0" \
    --type "SPARK" \
    --job-level-cost-allocation-configuration '{
        "enabled": false
    }'
```

Mettez à jour le `jobLevelCostAllocationConfiguration` paramètre d'une application existante :

```
# Enable job-level cost allocation:
aws emr-serverless update-application \
    --application-id <application-id> \
    --job-level-cost-allocation-configuration '{
        "enabled": true
    }'

# Disable job-level cost allocation:
aws emr-serverless update-application \
    --application-id <application-id> \
    --job-level-cost-allocation-configuration '{
        "enabled": false
    }'
```

Considérations et restrictions

- L'activation de la répartition des coûts au niveau des tâches n'attribue pas rétroactivement les coûts pour les séries de tâches terminées avant l'activation de la fonctionnalité. Les jobs démarrés après l'activation de la fonctionnalité seront soumis à une attribution des coûts granulaire.
- Le paramètre de répartition des coûts au niveau du travail ne peut être mis à jour que lorsqu'une application est à l'état CREATED ou STOPPED.

- Lorsque la répartition des coûts au niveau des tâches est activée, les coûts sont attribués à des séries de tâches individuelles plutôt qu'à l'application. Pour afficher les coûts agrégés au niveau de l'application, vous devez appliquer des balises cohérentes (telles que le nom de l'application ou l'identifiant de l'application) à toutes les exécutions de tâches au sein de cette application et filtrer en fonction de ces balises dans Cost Explorer ou Cost and Usage Reports.

Exécutez des charges de travail interactives avec EMR Serverless via EMR Studio

Avec les applications interactives EMR Serverless, exécutez des charges de travail interactives pour Spark avec EMR Serverless à l'aide de blocs-notes hébergés dans EMR Studio.

Présentation de

Une application interactive est une application EMR sans serveur dont les fonctionnalités interactives sont activées. Avec les applications interactives Amazon EMR Serverless, vous pouvez exécuter des charges de travail interactives avec des blocs-notes Jupyter gérés dans Amazon EMR Studio. Cela permet aux ingénieurs de données, aux scientifiques des données et aux analystes de données d'utiliser EMR Studio pour exécuter des analyses interactives avec des ensembles de données dans des magasins de données tels qu'Amazon S3 et Amazon DynamoDB.

Les cas d'utilisation des applications interactives dans EMR Serverless sont les suivants :

- Les ingénieurs de données utilisent l'expérience IDE d'EMR Studio pour créer un script ETL. Le script ingère des données sur site, les transforme à des fins d'analyse et les stocke dans Amazon S3.
- Les data scientists utilisent des carnets pour explorer les ensembles de données et entraîner des modèles d'apprentissage automatique (ML) afin de détecter les anomalies dans les ensembles de données.
- Les analystes de données explorent les ensembles de données et créent des scripts qui génèrent des rapports quotidiens pour mettre à jour des applications telles que les tableaux de bord commerciaux.

Conditions préalables

Pour utiliser des charges de travail interactives avec EMR Serverless, répondez aux exigences suivantes :

- Les applications interactives EMR Serverless sont prises en charge avec Amazon EMR 6.14.0 et versions ultérieures.

- Pour accéder à votre application interactive, exécuter les charges de travail que vous soumettez et exécuter des blocs-notes interactifs depuis EMR Studio, vous avez besoin d'autorisations et de rôles spécifiques. Pour plus d'informations, consultez Autorisations requises pour les charges de travail interactives.

Autorisations requises pour les charges de travail interactives

Outre les autorisations de base requises pour accéder à EMR Serverless, configurez des autorisations supplémentaires pour votre identité ou votre rôle IAM :

Pour accéder à votre application interactive

Configurez les autorisations d'utilisateur et d'espace de travail pour EMR Studio. Pour plus d'informations, consultez la section Configurer les autorisations utilisateur d'EMR Studio dans le guide de gestion Amazon EMR.

Pour exécuter les charges de travail que vous soumettez avec EMR Serverless

Configurez un rôle d'exécution des tâches. Pour plus d'informations, consultez Création d'un rôle d'exécution de tâches.

Pour exécuter les blocs-notes interactifs depuis EMR Studio

Ajoutez les autorisations supplémentaires suivantes à la politique IAM pour les utilisateurs de Studio :

- **`emr-serverless:AccessInteractiveEndpoints`**- Accorde l'autorisation d'accéder à l'application interactive que vous spécifiez et de vous y connecter`Resource`. Cette autorisation est requise pour se connecter à une application EMR Serverless depuis un espace de travail EMR Studio.
- **`iam:PassRole`**- Accorde l'autorisation d'accéder au rôle d'exécution IAM que vous prévoyez d'utiliser lorsque vous vous connectez à une application. L'`PassRole`autorisation appropriée est requise pour se connecter à une application EMR Serverless depuis un espace de travail EMR Studio.

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessInteractiveRole"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
    }
  ]
}
```

Configuration d'applications interactives

Suivez les étapes de haut niveau suivantes pour créer une application EMR sans serveur dotée de fonctionnalités interactives à partir d'Amazon EMR Studio dans le. AWS Management Console

1. Suivez les étapes décrites Commencer à utiliser Amazon EMR Serverless pour créer une application.
2. Lancez ensuite un espace de travail depuis EMR Studio et connectez-le à une application EMR Serverless en tant qu'option de calcul. Pour plus d'informations, reportez-vous à l'onglet Charge de travail interactive de l'étape 2 de la documentation de démarrage d'EMR Serverless.

Lorsque vous attachez une application à un espace de travail Studio, le démarrage de l'application se déclenche automatiquement si elle n'est pas déjà en cours d'exécution. Vous pouvez également prédémarrer l'application et la garder prête avant de l'associer à l'espace de travail.

Considérations relatives aux applications interactives

- Les applications interactives EMR Serverless sont prises en charge avec Amazon EMR 6.14.0 et versions ultérieures.
- EMR Studio est le seul client intégré aux applications interactives EMR Serverless. Les fonctionnalités EMR Studio suivantes ne sont pas prises en charge par les applications interactives EMR Serverless : collaboration dans l'espace de travail, SQL Explorer et exécution programmatique de blocs-notes.
- Les applications interactives ne sont prises en charge que pour le moteur Spark.
- Les applications interactives prennent en charge les noyaux Python 3 PySpark et Spark Scala.
- Vous pouvez exécuter jusqu'à 25 blocs-notes simultanément sur une seule application interactive.
- Aucun point de terminaison ou interface API ne prend en charge les blocs-notes Jupyter auto-hébergés dotés d'applications interactives.
- Pour une expérience de démarrage optimisée, nous vous suggérons de configurer la capacité préinitialisée pour les pilotes et les exécuteurs, et de démarrer votre application au préalable. Lorsque vous prédémarrez l'application, vous vous assurez qu'elle est prête lorsque vous souhaitez l'associer à votre espace de travail.

```
aws emr-serverless start-application \
--application-id your-application-id
```

- Par défaut, `autoStopConfig` est activé pour les applications. Cela arrête l'application après 30 minutes d'inactivité. Vous pouvez modifier cette configuration dans le cadre de votre `update-application` demande `create-application` ou de votre demande.
- Lorsque vous utilisez une application interactive, nous vous conseillons de configurer une capacité pré-initialisée de noyaux, de pilotes et d'exécuteurs pour exécuter vos ordinateurs portables. Chaque session interactive Spark nécessite un noyau et un pilote. EMR Serverless gère donc un programme de travail de noyau préinitialisé pour chaque pilote préinitialisé. Par défaut, EMR Serverless conserve une capacité préinitialisée d'un utilisateur du noyau dans l'ensemble de l'application, même si vous ne spécifiez aucune capacité préinitialisée pour les pilotes. Chaque kernel worker utilise 4 vCPU et 16 Go de mémoire. Pour obtenir des informations sur les prix actuels, consultez la page de tarification d'Amazon EMR.
- Vous devez disposer d'un quota de service vCPU suffisant Compte AWS pour exécuter des charges de travail interactives. Si vous n'exécutez pas de charges de travail compatibles avec

Lake Formation, nous vous conseillons d'utiliser au moins 24 vCPU. Si c'est le cas, nous vous conseillons d'utiliser au moins 28 vCPU.

- EMR Serverless arrête automatiquement les noyaux des ordinateurs portables s'ils sont inactifs depuis plus de 60 minutes. EMR Serverless calcule le temps d'inactivité du noyau depuis la dernière activité effectuée pendant la session du bloc-notes. Vous ne pouvez actuellement pas modifier le paramètre de délai d'inactivité du noyau.
- Pour activer Lake Formation avec des charges de travail interactives, définissez la configuration comme `spark.emr-serverless.lakeformation.enabled` étant inférieure à la `spark-defaults` classification dans l'`runtime-configuration`objet lorsque vous créez une `true` application EMR sans serveur. Pour en savoir plus, consultez Enabling Lake Formation in Amazon EMR.

Exécutez des charges de travail interactives avec EMR sans serveur via un point de terminaison Apache Livy

Avec les versions 6.14.0 et supérieures d'Amazon EMR, créez et activez un point de terminaison Apache Livy lors de la création d'une application EMR sans serveur et exécutez des charges de travail interactives via vos blocs-notes auto-hébergés ou avec un client personnalisé. Un point de terminaison Apache Livy offre les avantages suivants :

- Vous pouvez vous connecter en toute sécurité à un point de terminaison Apache Livy via les blocs-notes Jupyter et gérer les charges de travail Apache Spark avec l'interface REST d'Apache Livy.
- Utilisez les opérations de l'API REST d'Apache Livy pour les applications Web interactives qui utilisent les données des charges de travail Apache Spark.

Conditions préalables

Pour utiliser un point de terminaison Apache Livy avec EMR Serverless, répondez aux exigences suivantes :

- Suivez les étapes décrites dans Getting started with Amazon EMR Serverless.
- Pour exécuter des charges de travail interactives via les points de terminaison Apache Livy, vous avez besoin de certaines autorisations et de certains rôles. Pour plus d'informations, reportez-vous à la section Autorisations requises pour les charges de travail interactives.

Autorisations requises

Outre les autorisations requises pour accéder à EMR Serverless, ajoutez également les autorisations suivantes à votre rôle IAM pour accéder à un point de terminaison Apache Livy et exécuter des applications :

- `emr-serverless:AccessLivyEndpoints`— autorise l'accès et la connexion à l'application compatible Livy que vous spécifiez comme. `Resource` Vous avez besoin de cette autorisation pour exécuter les opérations d'API REST disponibles depuis le point de terminaison Apache Livy.
- `iam:PassRole`— accorde l'autorisation d'accéder au rôle d'exécution IAM lors de la création de la session Apache Livy. EMR Serverless utilisera ce rôle pour exécuter vos charges de travail.
- `emr-serverless:GetDashboardForJobRun`— autorise la génération de l'interface utilisateur de Spark Live et des liens vers les journaux des pilotes et donne accès aux journaux dans le cadre des résultats de la session Apache Livy.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessLivyEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessExecutionRole"
      ],
```

```
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
    },
    {
      "Sid": "EMRServerlessDashboardAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetDashboardForJobRun"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    }
  ]
}
```

Prise en main

Pour créer une application compatible Apache Livy et l'exécuter, procédez comme suit.

1. Pour créer une application compatible avec Apache Livy, exécutez la commande suivante.

```
 aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
 --release-label <Amazon EMR-release-version>
--interactive-configuration '{"livyEndpointEnabled": true}'
```

2. Une fois que EMR Serverless a créé votre application, démarrez-la pour rendre le point de terminaison Apache Livy disponible.

```
aws emr-serverless start-application \
--application-id application-id
```

   Utilisez la commande suivante pour vérifier l'état de votre demande. Une fois le statut atteint`STARTED`, accédez au point de terminaison Apache Livy.

```
aws emr-serverless get-application \
```

```
--region <AWS_REGION> --application-id >application_id>
```

3. Utilisez l'URL suivante pour accéder au point de terminaison :

```
https://_<application-id>_.livy.emr-serverless-
services._<AWS_REGION>_.amazonaws.com
```

Une fois que le terminal est prêt, soumettez les charges de travail en fonction de votre cas d'utilisation. Vous devez signer chaque demande envoyée au point de terminaison avec le SIGv4 protocole et transmettre un en-tête d'autorisation. Vous pouvez utiliser les méthodes suivantes pour exécuter des charges de travail :

- Client HTTP : soumettez les opérations de l'API de votre point de terminaison Apache Livy avec un client HTTP personnalisé.
- Noyau Sparkmagic — exécutez le noyau Sparkmagic localement et soumettez des requêtes interactives avec les blocs-notes Jupyter.

Clients HTTP

Pour créer une session Apache Livy, soumettez-la `emr-serverless.session.executionRoleArn` dans les `conf` paramètres du corps de votre requête. L'exemple suivant est un exemple de `POST /sessions` demande.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "<executionRoleArn>"
    }
}
```

Le tableau suivant décrit toutes les opérations d'API Apache Livy disponibles.

Opération API	Description
OBTENIR /sessions	Renvoie la liste de toutes les sessions interactives actives.

Opération API	Description
POST/sessions	Crée une nouvelle session interactive via Spark ou Pyspark.
OBTENEZ /sessions/ < > *sessionId*	Renvoie les informations de session.
GET /sessions/ < >/state *sessionId*	Renvoie l'état de la session.
SUPPRIMER /sessions/ < > *sessionId*	Arrête et supprime la session.
GET /sessions/ < >/déclarations *sessionId*	Renvoie toutes les instructions d'une session.
POST /sessions/ < >/déclarations *sessionId*	Exécute une instruction au cours d'une session.
GET /sessions/ < >/déclarations/< > *sessionId statementId*	Renvoie les détails de l'instruction spécifiée dans une session.
POST /sessions/ < >/déclarations/< >/annuler *sessionId statementId*	Annule l'instruction spécifiée dans cette session.

Envoi de requêtes au point de terminaison Apache Livy

Vous pouvez également envoyer des demandes directement au point de terminaison Apache Livy à partir d'un client HTTP. Cela vous permet d'exécuter du code à distance pour vos cas d'utilisation en dehors d'un bloc-notes.

Avant de commencer à envoyer des demandes au terminal, assurez-vous d'avoir installé les bibliothèques suivantes :

```
pip3 install botocore awscrt requests
```

Voici un exemple de script Python permettant d'envoyer des requêtes HTTP directement à un point de terminaison :

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
import botocore.session
```

```
import json, pprint, textwrap

endpoint = 'https://<application_id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com'
headers = {'Content-Type': 'application/json'}

session = botocore.session.Session()
signer = crt.auth.CrtS3SigV4Auth(session.get_credentials(), 'emr-serverless',
 '<AWS_REGION>')


### Create session request

data = {'kind': 'pyspark', 'heartbeatTimeoutInSecond': 60, 'conf': { 'emr-
serverless.session.executionRoleArn': 'arn:aws:iam::123456789012:role/role1'}}

request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r.json())


### List Sessions Request

request = AWSRequest(method='GET', url=endpoint + "/sessions", headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r2 = requests.get(prepped.url, headers=prepped.headers)
pprint.pprint(r2.json())
```

```
### Get session state

session_url = endpoint + r.headers['location']

request = AWSRequest(method='GET', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r3 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r3.json())


### Submit Statement

data = {
      'code': "1 + 1"
}

statements_url = endpoint + r.headers['location'] + "/statements"

request = AWSRequest(method='POST', url=statements_url, data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r4 = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r4.json())

### Check statements results

specific_statement_url = endpoint + r4.headers['location']

request = AWSRequest(method='GET', url=specific_statement_url, headers=headers)
```

```
request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r5 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r5.json())


### Delete session


session_url = endpoint + r.headers['location']

request = AWSRequest(method='DELETE', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r6 = requests.delete(prepped.url, headers=prepped.headers)

pprint.pprint(r6.json())
```

Noyau Sparkmagic

Avant d'installer sparkmagic, assurez-vous d'avoir configuré les AWS informations d'identification dans l'instance dans laquelle vous souhaitez installer sparkmagic

1. Installez sparkmagic en suivant les étapes d'installation. Notez que vous n'effectuez que les quatre premières étapes.
2. Le noyau sparkmagic prend en charge les authentificateurs personnalisés. Vous pouvez donc intégrer un authentificateur au noyau sparkmagic afin que chaque demande soit signée. SIGv4
3. Installez l'authentificateur personnalisé EMR Serverless.

   ```
   pip install emr-serverless-customauth
   ```

4. Indiquez maintenant le chemin d'accès à l'authentificateur personnalisé et l'URL du point de terminaison Apache Livy dans le fichier json de configuration de sparkmagic. Utilisez la commande suivante pour ouvrir le fichier de configuration.

```
vim ~/.sparkmagic/config.json
```

Voici un exemple de `config.json` fichier.

```
{
"kernel_python_credentials" : {
    "username": "",
    "password": "",
    "url": "https://<application-id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com",
    "auth": "Custom_Auth"
  },

  "kernel_scala_credentials" : {
    "username": "",
    "password": "",
    "url": "https://<application-id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com",
    "auth": "Custom_Auth"
  },
  "authenticators": {
    "None": "sparkmagic.auth.customauth.Authenticator",
    "Basic_Access": "sparkmagic.auth.basic.Basic",
    "Custom_Auth":
 "emr_serverless_customauth.customauthenticator.EMRServerlessCustomSigV4Signer"
  },
  "livy_session_startup_timeout_seconds": 600,
  "ignore_ssl_errors": false
}
```

5. Démarrez Jupyter Lab. Il doit utiliser l'authentification personnalisée que vous avez configurée lors de la dernière étape.
6. Vous pouvez ensuite exécuter les commandes de bloc-notes suivantes et votre code pour commencer.

```
%%info //Returns the information about the current sessions.
```

```
%%configure -f //Configure information specific to a session. We supply
 executionRoleArn in this example. Change it for your use case.
{
    "driverMemory": "4g",
    "conf": {
          "emr-serverless.session.executionRoleArn":
 "arn:aws:iam::123456789012:role/JobExecutionRole"
    }
}
```

```
<your code>//Run your code to start the session
```

En interne, chaque instruction appelle chacune des opérations de l'API Apache Livy via l'URL du point de terminaison Apache Livy configurée. Vous pouvez ensuite rédiger vos instructions en fonction de votre cas d'utilisation.

Considérations

Tenez compte des considérations suivantes lorsque vous exécutez des charges de travail interactives via les points de terminaison Apache Livy.

- EMR Serverless maintient l'isolation au niveau de la session en utilisant le principal de l'appelant. Le principal appelant qui crée la session est le seul à pouvoir accéder à cette session. Pour une isolation plus précise, configurez une identité source lorsque vous utilisez des informations d'identification. Dans ce cas, EMR Serverless applique une isolation au niveau de la session en fonction du principal appelant et de l'identité de la source. Pour plus d'informations sur l'identité de la source, reportez-vous à la section Surveiller et contrôler les actions entreprises avec des rôles assumés.
- Les points de terminaison Apache Livy sont pris en charge avec les versions 6.14.0 et supérieures d'EMR Serverless.
- Les points de terminaison Apache Livy ne sont pris en charge que pour le moteur Apache Spark.
- Les points de terminaison Apache Livy prennent en charge Scala Spark et. PySpark
- Par défaut, `autoStopConfig` est activé dans vos applications. Cela signifie que les applications s'arrêtent après 15 minutes d'inactivité. Vous pouvez modifier cette configuration dans le cadre de votre `update-application` demande `create-application` ou de votre demande.

- Vous pouvez exécuter jusqu'à 25 sessions simultanées sur une seule application compatible avec le point de terminaison Apache Livy.
- Pour une expérience de démarrage optimale, nous vous conseillons de configurer la capacité pré-initialisée pour les pilotes et les exécuteurs.
- Vous devez démarrer manuellement votre application avant de vous connecter au point de terminaison Apache Livy.
- Vous devez disposer d'un quota de service vCPU suffisant Compte AWS pour exécuter des charges de travail interactives avec le point de terminaison Apache Livy. Nous vous conseillons d'utiliser au moins 24 vCPU.
- Le délai d'expiration de session Apache Livy par défaut est de 1 heure. Si vous n'exécutez aucune instruction pendant une heure, Apache Livy supprime la session et libère le pilote et les exécuteurs. À partir de la version emr-7.8.0, cette valeur peut être définie en spécifiant le `ttl` paramètre dans le cadre de la `/sessions POST` demande Livy, par exemple `2h` (heures), (minutes), `120m` (secondes), `7200s` (millisecondes). `7200000ms`

  > **Note**
  >
  > Cette configuration ne peut pas être modifiée avant emr-7.8.0. Voici un exemple de corps de `POST /sessions` demande.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "executionRoleArn"
    },
    "ttl": "2h"
}
```

- À partir de la version emr-7.8.0 d'Amazon EMR pour les applications dont le contrôle d'accès détaillé est LakeFormation activé, le paramètre peut être désactivé par session. Pour plus d'informations sur l'activation du contrôle d'accès détaillé pour une application EMR sans serveur, reportez-vous à la section Méthodes de contrôle d'accès détaillé.

> **Note**
>
> Lake Formation ne peut pas être activée pour une session alors qu'elle n'a pas été activée pour une application. Voici un exemple de corps de `POST /sessions` demande.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "executionRoleArn"
    },
    "spark.emr-serverless.lakeformation.enabled" : "false"
}
```

- Seules les sessions actives peuvent interagir avec un point de terminaison Apache Livy. Une fois la session terminée, annulée ou terminée, vous ne pouvez pas y accéder via le point de terminaison Apache Livy.

Journalisation et surveillance

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances des applications et des tâches EMR sans serveur. Vous devez collecter des données de surveillance provenant de tous les composants de vos solutions EMR Serverless afin de pouvoir corriger plus facilement les défaillances multipoints si elles se produisent.

Rubriques

Stockage des journaux

Pour suivre l'avancement de vos tâches sur EMR Serverless et résoudre les problèmes d'échec, choisissez la manière dont EMR Serverless stocke et diffuse les journaux des applications. Lorsque vous soumettez une exécution de tâche, spécifiez le stockage géré, Amazon S3 et Amazon CloudWatch comme options de journalisation.

Avec CloudWatch, spécifiez les types de journaux et les emplacements des journaux que vous souhaitez utiliser, ou acceptez les types et emplacements par défaut. Pour plus d'informations sur CloudWatch les journaux, reportez-vous àthe section called “Amazon CloudWatch”. En ce qui concerne le stockage géré et la journalisation S3, le tableau suivant répertorie les emplacements des journaux et la disponibilité de l'interface utilisateur auxquels vous pouvez vous attendre si vous choisissez le stockage géré, les compartiments Amazon S3 ou les deux.

Option	Journaux d'événements	Journaux de conteneurs	Interface utilisateur de l'application
Stockage géré	Stocké dans un espace de stockage géré	Stocké dans un espace de stockage géré	Pris en charge

Option	Journaux d'événements	Journaux de conteneurs	Interface utilisateur de l'application
Stockage géré et compartiment S3	Stocké aux deux endroits	Stocké dans un compartiment S3	Pris en charge
Compartiment Amazon S3	Stocké dans un compartiment S3	Stocké dans un compartiment S3	Non pris en charge [1]

[1] Nous vous conseillons de conserver l'option Stockage géré sélectionnée. Dans le cas contraire, vous ne pouvez pas utiliser l'application intégrée UIs.

Journalisation pour EMR sans serveur avec stockage géré

Par défaut, EMR Serverless stocke les journaux des applications en toute sécurité dans le stockage géré par Amazon EMR pendant une durée maximale de 30 jours.

> **Note**
>
> Si vous désactivez l'option par défaut, Amazon EMR ne pourra pas résoudre les problèmes liés à vos tâches en votre nom. Exemple : vous ne pouvez pas accéder à Spark-UI depuis la console EMR Serverless.

Pour désactiver cette option dans EMR Studio, décochez la case Autoriser AWS à conserver les journaux pendant 30 jours dans la section Paramètres supplémentaires de la page Soumettre la tâche.

Pour désactiver cette option depuis le AWS CLI, utilisez la `managedPersistenceMonitoringConfiguration` configuration lorsque vous soumettez une exécution de tâche.

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }
```

```
}
```

Si votre application EMR Serverless se trouve dans un sous-réseau privé avec des points de terminaison VPC pour Amazon S3 et que vous attachez une politique de point de terminaison pour contrôler l'accès, ajoutez les autorisations suivantes pour qu'EMR Serverless stocke et diffuse les journaux des applications. `Resource`Remplacez-les par les `AppInfo` compartiments du tableau des régions disponibles dans Exemples de politiques pour les sous-réseaux privés qui accèdent à Amazon S3.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessManagedLogging",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::prod.us-east-1.appinfo.src",
        "arn:aws:s3:::prod.us-east-1.appinfo.src/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalServiceName": "emr-serverless.amazonaws.com",
          "aws:SourceVpc": "vpc-12345678"
        }
      }
    }
  ]
}
```

En outre, utilisez la clé de `aws:SourceVpc` condition pour vous assurer que la demande transite par le VPC auquel le point de terminaison du VPC est attaché.

Journalisation pour EMR sans serveur avec des compartiments Amazon S3

Avant que vos tâches puissent envoyer des données de journal à Amazon S3, incluez les autorisations suivantes dans la politique d'autorisation pour le 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"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

Pour configurer un compartiment Amazon S3 afin de stocker les journaux provenant du AWS CLI, utilisez la `s3MonitoringConfiguration` configuration lorsque vous lancez une tâche. Pour ce faire, fournissez les éléments suivants `--configuration-overrides` dans la configuration.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/"
        }
    }
}
```

Pour les tâches par lots pour lesquelles aucune nouvelle tentative n'est activée, EMR Serverless envoie les journaux au chemin suivant :

```
'/applications/<applicationId>/jobs/<jobId>'
```

Les journaux du pilote Spark sont stockés dans le chemin suivant par EMR Serverless

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_DRIVER/'
```

Les journaux de l'exécuteur Spark sont stockés dans le chemin suivant par EMR Serverless

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_EXECUTOR/<EXECUTOR-ID>'
```

Le <EXECUTOR-ID>est un entier.

Les versions 7.1.0 et supérieures d'EMR Serverless prennent en charge les nouvelles tentatives pour les tâches de streaming et les tâches par lots. Si vous exécutez une tâche avec les nouvelles tentatives activées, EMR Serverless ajoute automatiquement un numéro de tentative au préfixe du chemin du journal, afin que vous puissiez mieux distinguer et suivre les journaux.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'
```

Journalisation pour EMR sans serveur avec Amazon CloudWatch

Lorsque vous soumettez une tâche à une application EMR Serverless, choisissez Amazon CloudWatch comme option pour stocker les journaux de votre candidature. Cela vous permet d'utiliser les fonctionnalités d'analyse des CloudWatch CloudWatch journaux telles que Logs Insights et Live Tail. Vous pouvez également diffuser des journaux depuis d'autres systèmes, par exemple CloudWatch à des OpenSearch fins d'analyse plus approfondie.

EMR Serverless fournit une journalisation en temps réel pour les journaux des conducteurs. Vous pouvez accéder aux journaux en temps réel grâce à la fonctionnalité CloudWatch Live Tail ou via les commandes de queue de la CloudWatch CLI.

Par défaut, la CloudWatch journalisation est désactivée pour EMR Serverless. Pour l'activer, utilisez la configuration dansAWS CLI.

> **Note**
>
> Amazon CloudWatch publie des journaux en temps réel, ce qui lui permet de mobiliser davantage de ressources de la part de ses employés. Si vous optez pour une faible capacité

de travail, l'impact sur la durée d'exécution de votre travail peut augmenter. Si vous activez la CloudWatch journalisation, nous vous suggérons de choisir une capacité de travail supérieure. Il est également possible que la publication du journal soit ralentie si le taux de transactions par seconde (TPS) est trop faible pour. `PutLogEvents` La configuration CloudWatch de régulation est globale pour tous les services, y compris EMR Serverless. Pour plus d'informations, reportez-vous à la section Comment déterminer la régulation dans mes CloudWatch journaux ? sur AWS re:post.

Autorisations requises pour se connecter avec CloudWatch

Avant que vos tâches puissent envoyer des données de journal à Amazon CloudWatch, incluez les autorisations suivantes dans la politique d'autorisation pour le rôle d'exécution des tâches.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:*"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSPutlogevents"
```

```
    }
  ]
}
```

AWS CLI

Pour configurer Amazon CloudWatch afin qu'il stocke les journaux pour EMR Serverless à partir du AWS CLI, utilisez la `cloudWatchLoggingConfiguration` configuration lorsque vous lancez une tâche. Pour ce faire, fournissez les remplacements de configuration suivants. Vous pouvez également fournir un nom de groupe de journaux, un nom de préfixe de flux de journaux, des types de journaux et un ARN de clé de chiffrement.

Si vous ne spécifiez pas de valeurs facultatives, CloudWatch publie les journaux dans un groupe de journaux par défaut`/aws/emr-serverless`, avec le flux de journaux par défaut`/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`*.

Les versions 7.1.0 et supérieures d'EMR Serverless prennent en charge les nouvelles tentatives pour les tâches de streaming et les tâches par lots. Si vous avez activé les nouvelles tentatives pour une tâche, EMR Serverless ajoute automatiquement un numéro de tentative au préfixe du chemin du journal, afin que vous puissiez mieux distinguer et suivre les journaux.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/worker-type'
```

Voici la configuration minimale requise pour activer la CloudWatch journalisation Amazon avec les paramètres par défaut pour EMR Serverless :

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true
         }
     }
}
```

L'exemple suivant montre toutes les configurations obligatoires et facultatives qui spécifient le moment où vous activez la CloudWatch journalisation Amazon pour EMR Serverless. Les `logTypes` valeurs prises en charge sont également répertoriées dans l'exemple suivant.

```
{
```

```
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true, // Required
            "logGroupName": "Example_logGroup", // Optional
            "logStreamNamePrefix": "Example_logStream", // Optional
            "encryptionKeyArn": "key-arn", // Optional
            "logTypes": {
                "SPARK_DRIVER": ["stdout", "stderr"] //List of values
             }
         }
     }
}
```

Par défaut, EMR Serverless publie uniquement le pilote stdout et stderr vers lequel stderr se connecte. CloudWatch Si vous souhaitez d'autres journaux, spécifiez un rôle de conteneur et les types de journaux correspondants `logTypes` dans le champ.

La liste suivante indique les types de travailleurs pris en charge qui sont spécifiés pour la `logTypes` configuration :

Spark

- `SPARK_DRIVER : ["STDERR", "STDOUT"]`
- `SPARK_EXECUTOR : ["STDERR", "STDOUT"]`

Hive

- `HIVE_DRIVER : ["STDERR", "STDOUT", "HIVE_LOG", "TEZ_AM"]`
- `TEZ_TASK : ["STDERR", "STDOUT", "SYSTEM_LOGS"]`

Bûches rotatives

Amazon EMR Serverless peut alterner les journaux d'applications et les journaux d'événements Spark. La rotation des journaux permet de résoudre le problème des tâches de longue durée qui génèrent des fichiers journaux volumineux qui peuvent occuper tout l'espace disque. La rotation des journaux vous permet d'économiser de l'espace de stockage sur disque et de réduire le nombre d'échecs de tâches, car il ne vous reste plus d'espace sur votre disque.

La rotation des journaux est activée par défaut et n'est disponible que pour les tâches Spark.

Journaux d'événements Spark

> **Note**
>
> La rotation du journal d'événements Spark est disponible sur toutes les étiquettes de version d'Amazon EMR.

Au lieu de générer un seul fichier journal des événements, EMR Serverless fait pivoter le journal des événements à intervalles réguliers et supprime les anciens fichiers journaux d'événements. La rotation des journaux n'affecte pas les journaux chargés dans le compartiment S3.

Journaux des applications Spark

> **Note**
>
> La rotation des journaux des applications Spark est disponible sur toutes les étiquettes de version d'Amazon EMR.

EMR Serverless fait également pivoter les journaux de l'application Spark pour les pilotes et les exécuteurs, tels que les fichiers et. `stdout` `stderr` Vous pouvez accéder aux derniers fichiers journaux en choisissant les liens des journaux dans Studio à l'aide des liens Spark History Server et Live UI. Les fichiers journaux sont les versions tronquées des derniers journaux. Pour faire référence aux anciens journaux soumis à une rotation, spécifiez un emplacement Amazon S3 lors du stockage des journaux. Reportez-vous à la section Journalisation pour EMR sans serveur avec des compartiments Amazon S3 pour plus d'informations.

Vous trouverez les derniers fichiers journaux à l'emplacement suivant. EMR Serverless actualise les fichiers toutes les 15 secondes. Ces fichiers peuvent aller de 0 Mo à 128 Mo.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/stderr.gz
```

L'emplacement suivant contient les anciens fichiers ayant fait l'objet d'une rotation. La taille de chaque fichier est de 128 Mo.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/archived/
stderr_<index>.gz
```

Le même comportement s'applique également aux exécuteurs Spark. Cette modification s'applique uniquement à la journalisation S3. La rotation des journaux n'apporte aucune modification aux flux de journaux chargés sur Amazon CloudWatch.

Les versions 7.1.0 et supérieures d'EMR Serverless prennent en charge les nouvelles tentatives pour le streaming et les tâches par lots. Si vous avez activé les nouvelles tentatives pour votre tâche, EMR Serverless ajoute un préfixe au chemin du journal pour ces tâches afin que vous puissiez mieux suivre et distinguer les journaux les uns des autres. Ce chemin contient tous les journaux ayant fait l'objet d'une rotation.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

Chiffrement des journaux

Chiffrement des journaux EMR sans serveur avec stockage géré

Pour chiffrer les journaux du stockage géré à l'aide de votre propre clé KMS, utilisez la `managedPersistenceMonitoringConfiguration` configuration lorsque vous soumettez une exécution de tâche.

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration" : {
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Chiffrement des journaux EMR sans serveur avec des compartiments Amazon S3

Pour chiffrer les journaux de votre compartiment Amazon S3 avec votre propre clé KMS, utilisez la `s3MonitoringConfiguration` configuration lorsque vous soumettez une exécution de tâche.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/",
```

```
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Chiffrer les journaux EMR sans serveur avec Amazon CloudWatch

Pour chiffrer les journaux dans Amazon CloudWatch avec votre propre clé KMS, utilisez la `cloudWatchLoggingConfiguration` configuration lorsque vous soumettez une exécution de tâche.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true,
            "encryptionKeyArn": "key-arn"
         }
     }
}
```

Autorisations requises pour le chiffrement des journaux

Dans cette section

- Autorisations utilisateur requises
- Autorisations relatives aux clés de chiffrement pour Amazon S3 et le stockage géré
- Autorisations relatives aux clés de chiffrement pour Amazon CloudWatch

Autorisations utilisateur requises

L'utilisateur qui soumet le travail ou consulte les journaux ou l'application UIs doit être autorisé à utiliser la clé. Vous pouvez spécifier les autorisations dans la politique clé KMS ou dans la stratégie IAM pour l'utilisateur, le groupe ou le rôle. Si l'utilisateur qui soumet la tâche ne dispose pas des autorisations clés KMS, EMR Serverless rejette la soumission de l'exécution de la tâche.

Exemple de politique clé

La politique clé suivante fournit les autorisations nécessaires pour `kms:GenerateDataKey` et `kms:Decrypt` :

```
{
    "Effect": "Allow",
    "Principal":{
       "AWS": "arn:aws:iam::111122223333:user/user-name"
     },
     "Action": [
       "kms:GenerateDataKey",
       "kms:Decrypt"
      ],
     "Resource": "*"
 }
```

Exemple de politique IAM

La politique IAM suivante fournit les autorisations nécessaires pour `kms:GenerateDataKey` et `kms:Decrypt` :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Pour lancer l'interface utilisateur Spark ou Tez, autorisez vos utilisateurs, groupes ou rôles à accéder à l'`emr-serverless:GetDashboardForJobRun`API comme suit :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetDashboardForJobRun"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRSERVERLESSGetdashboardforjobrun"
    }
  ]
}
```

Autorisations relatives aux clés de chiffrement pour Amazon S3 et le stockage géré

Lorsque vous chiffrez des journaux avec votre propre clé de chiffrement dans le stockage géré ou dans vos compartiments S3, configurez les autorisations relatives aux clés KMS comme suit.

Le `emr-serverless.amazonaws.com` principal doit disposer des autorisations suivantes dans la politique relative à la clé KMS :

```
{
    "Effect": "Allow",
    "Principal":{
       "Service": "emr-serverless.amazonaws.com"
     },
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey"
      ],
     "Resource": "*"
     "Condition": {
       "StringLike": {
         "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
       }
```

```
    }
}
```

Pour des raisons de sécurité, nous vous suggérons d'ajouter une clé de `aws:SourceArn` condition à la politique de clé KMS. La clé de condition globale IAM `aws:SourceArn` permet de garantir qu'EMR Serverless utilise la clé KMS uniquement pour l'ARN d'une application.

Le rôle d'exécution des tâches doit disposer des autorisations suivantes dans sa politique IAM :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Autorisations relatives aux clés de chiffrement pour Amazon CloudWatch

Pour associer l'ARN de la clé KMS à votre groupe de journaux, utilisez la politique IAM suivante pour le rôle d'exécution des tâches.

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "logs:AssociateKmsKey"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSAssociatekmskey"
    }
  ]
}
```

Configurez la politique relative aux clés KMS pour accorder des autorisations KMS à Amazon CloudWatch :

JSON

```
{
  "Version":"2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:*:123456789012:*"
        }
      },
      "Sid": "AllowKMSDecrypt"
    }
  ]
}
```

Configuration des propriétés Apache Log4j2 pour Amazon EMR Serverless

Cette page décrit comment configurer les propriétés personnalisées d'Apache Log4j 2.x pour les tâches EMR sans serveur sur. `StartJobRun` Si vous souhaitez configurer les classifications Log4j au niveau de l'application, reportez-vous à. Configuration d'application par défaut pour EMR Serverless

Configuration des propriétés de Spark Log4j2 pour Amazon EMR Serverless

Avec les versions 6.8.0 et supérieures d'Amazon EMR, vous pouvez personnaliser les propriétés d'Apache Log4j 2.x pour spécifier des configurations de journal précises. Cela simplifie le dépannage de vos tâches Spark sur EMR Serverless. Pour configurer ces propriétés, utilisez les `spark-executor-log4j2` classifications `spark-driver-log4j2` et.

Rubriques

- Classifications Log4j2 pour Spark
- Exemple de configuration Log4j2 pour Spark
- Log4j2 dans les exemples de tâches Spark
- Considérations relatives à Log4j2 pour Spark

Classifications Log4j2 pour Spark

Pour personnaliser les configurations du journal Spark, utilisez les classifications suivantes avec `applicationConfiguration`. Pour configurer les propriétés de Log4j 2.x, utilisez ce qui suit. `properties`

`spark-driver-log4j2`

Cette classification définit les valeurs du `log4j2.properties` fichier pour le pilote.

`spark-executor-log4j2`

Cette classification définit les valeurs du `log4j2.properties` fichier pour l'exécuteur.

Exemple de configuration Log4j2 pour Spark

L'exemple suivant montre comment soumettre une tâche Spark pour personnaliser les `applicationConfiguration` configurations Log4j2 pour le pilote et l'exécuteur Spark.

Pour configurer les classifications Log4j au niveau de l'application plutôt que lorsque vous soumettez le travail, reportez-vous à. Configuration d'application par défaut pour EMR Serverless

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
    }'
    --configuration-overrides '{
        "applicationConfiguration": [
             {
                "classification": "spark-driver-log4j2",
                "properties": {
                    "rootLogger.level":"error", // will only display Spark error logs
                    "logger.IdentifierForClass.name": "classpath for setting logger",
                    "logger.IdentifierForClass.level": "info"

                }
            },
            {
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level":"error", // will only display Spark error logs
                    "logger.IdentifierForClass.name": "classpath for setting logger",
                    "logger.IdentifierForClass.level": "info"
                }
            }
       ]
    }'
```

Log4j2 dans les exemples de tâches Spark

Les exemples de code suivants montrent comment créer une application Spark pendant que vous initialisez une configuration Log4j2 personnalisée pour l'application.

Python

Example- Utiliser Log4j2 pour une tâche Spark avec Python

```
import os
import sys

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

app_name = "PySparkApp"
if __name__ == "__main__":
    spark = SparkSession\
        .builder\
        .appName(app_name)\
        .getOrCreate()

    spark.sparkContext._conf.getAll()
    sc = spark.sparkContext
    log4jLogger = sc._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger(app_name)

    LOGGER.info("pyspark script logger info")
    LOGGER.warn("pyspark script logger warn")
    LOGGER.error("pyspark script logger error")

    // your code here

    spark.stop()
```

Pour personnaliser Log4j2 pour le pilote lorsque vous exécutez une tâche Spark, utilisez la configuration suivante :

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
```

```
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example- Utilisation de Log4j2 pour une tâche Spark avec Scala

```
import org.apache.log4j.Logger
import org.apache.spark.sql.SparkSession

object ExampleClass {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
    .builder
    .appName(this.getClass.getName)
    .getOrCreate()

    val logger = Logger.getLogger(this.getClass);
    logger.info("script logging info logs")
    logger.warn("script logging warn logs")
    logger.error("script logging error logs")

// your code here
    spark.stop()
  }
}
```

Pour personnaliser Log4j2 pour le pilote lorsque vous exécutez une tâche Spark, utilisez la configuration suivante :

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.ExampleClass.level": "info",
          "logger.ExampleClass.name": "ExampleClass"
      }
}
```

Considérations relatives à Log4j2 pour Spark

Les propriétés Log4j2.x suivantes ne sont pas configurables pour les processus Spark :

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

Pour des informations détaillées sur les propriétés de Log4j2.x configurables, reportez-vous au fichier sur. `log4j2.properties.template` GitHub

Surveillance de l'EMR sans serveur

Cette section décrit les méthodes de surveillance de vos applications et tâches Amazon EMR Serverless.

Rubriques

- Surveillance des applications et des tâches EMR sans serveur
- Surveillez les métriques de Spark avec Amazon Managed Service pour Prometheus
- Mesures d'utilisation EMR sans serveur

Surveillance des applications et des tâches EMR sans serveur

Avec Amazon CloudWatch Metrics for EMR Serverless, vous pouvez recevoir des CloudWatch métriques en une minute et accéder à des CloudWatch tableaux de bord pour accéder aux near-real-time opérations et aux performances de vos applications EMR Serverless.

EMR Serverless envoie des métriques à chaque minute. CloudWatch EMR Serverless émet ces métriques au niveau de l'application ainsi qu'au niveau de la tâche, du type de travailleur et au niveau de l'application. capacity-allocation-type

Pour commencer, utilisez le modèle de tableau de CloudWatch bord EMR Serverless fourni dans le référentiel EMR GitHub Serverless et déployez-le.

> **Note**
>
> Les charges de travail interactives EMR sans serveur sont uniquement activées pour la surveillance au niveau de l'application et ont une nouvelle dimension de type de travailleur. `Spark_Kernel` Pour surveiller et déboguer vos charges de travail interactives, accédez aux journaux et à l'interface utilisateur d'Apache Spark depuis votre espace de travail EMR Studio.

Surveillance des métriques

> **Important**
>
> Nous restructurons notre affichage des métriques pour ajouter `ApplicationName` et `JobName` en tant que dimensions. Pour les versions 7.10 et ultérieures, les anciennes métriques ne seront plus mises à jour. Pour les versions EMR inférieures à 7.10, les anciennes métriques sont toujours disponibles.

Dimensions actuelles

Le tableau ci-dessous décrit les dimensions EMR Serverless disponibles dans l'espace de noms. `AWS/EMR Serverless`

Dimensions des métriques EMR sans serveur

Dimension	Description	
`ApplicationId`	Filtre toutes les métriques d'une application EMR sans serveur à l'aide de l'ID de l'application.	
`ApplicationName`	Filtre toutes les métriques d'une application EMR sans serveur utilisant le nom. Si le nom n'est pas fourni ou contient des caractères non	

Dimension	Description
	ASCII, il est publié sous le nom [Non spécifié].
`JobId`	Filtre toutes les métriques d'un EMR Serverless, l'ID d'exécuti on de la tâche.
`JobName`	Filtres pour toutes les métriques d'une tâche EMR sans serveur exécutée en utilisant le nom. Si le nom n'est pas fourni ou contient des caractères non ASCII, il est publié sous le nom [Non spécifié].
`WorkerType`	Filtres pour tous les indicateu rs d'un type de travailleur donné. Par exemple, vous pouvez filtrer pour `SPARK_DRI VER` et `SPARK_EXECUTORS` pour les tâches Spark.
`CapacityAllocation Type`	Filtres pour tous les indicateu rs d'un type d'allocation de capacité donné. Par exemple, vous pouvez filtrer `PreInitCapacity` pour la capacité pré-initialisée et `OnDemandCapacity` pour tout le reste.

Surveillance au niveau de l'application

Vous pouvez surveiller l'utilisation de la capacité au niveau de l'application EMR Serverless à l'aide des métriques Amazon. CloudWatch Vous pouvez également configurer un écran unique pour surveiller l'utilisation de la capacité des applications dans un CloudWatch tableau de bord.

Mesures relatives aux applications EMR sans serveur

Métrique	Description	Unit	Dimension
`MaxCPUAllowed`	Processeur maximal autorisé pour l'application.	vCPU	`ApplicationId` , `ApplicationName`
`MaxMemoryAllowed`	Mémoire maximale autorisée pour l'application, exprimée en Go.	Gigaoctets (Go)	`ApplicationId` , `ApplicationName`
`MaxStorageAllowed`	Stockage maximal en Go autorisé pour l'application.	Gigaoctets (Go)	`ApplicationId` , `ApplicationName`
`CPUAllocated`	Le nombre total de v CPUs alloués.	vCPU	`ApplicationId` , `ApplicationName` , `WorkerType` , `CapacityAllocationType`
`IdleWorkerCount`	Nombre total de travailleurs inactifs.	Nombre	`ApplicationId` , `ApplicationName` , `WorkerType` , `CapacityAllocationType`
`MemoryAllocated`	Mémoire totale allouée en Go.	Gigaoctets (Go)	`ApplicationId` , `ApplicationName` , `WorkerTyp`

Métrique	Description	Unit	Dimension
			`e` , `CapacityA llocationType`
`PendingCr eationWor kerCount`	Nombre total de travailleurs en attente de création.	Nombre	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`RunningWo rkerCount`	Nombre total de travailleurs utilisés par l'application.	Nombre	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`StorageAl located`	Stockage sur disque total en Go alloué.	Gigaoctets (Go)	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`TotalWork erCount`	Le nombre total de travailleurs disponibl es.	Nombre	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`

Surveillance au niveau des tâches

Amazon EMR Serverless envoie les métriques suivantes au niveau des tâches toutes les minutes. Amazon CloudWatch Vous pouvez accéder aux valeurs métriques pour les exécutions de tâches agrégées par état d'exécution des tâches. L'unité de chacune des métriques est le nombre.

Mesures EMR Serverless au niveau des tâches

Métrique	Description	Dimension
`SubmittedJobs`	Le nombre de tâches dans un état Soumis.	`ApplicationId` , `ApplicationName`
`PendingJobs`	Le nombre de tâches en attente.	`ApplicationId` , `ApplicationName`
`ScheduledJobs`	Le nombre de tâches dans un état planifié.	`ApplicationId` , `ApplicationName`
`RunningJobs`	Nombre de tâches en cours d'exécution.	`ApplicationId` , `ApplicationName`
`SuccessJobs`	Le nombre d'emplois dont l'état est « Success ».	`ApplicationId` , `ApplicationName`
`FailedJobs`	Le nombre de tâches en état d'échec.	`ApplicationId` , `ApplicationName`
`CancellingJobs`	Le nombre d'emplois dans un État annulable.	`ApplicationId` , `ApplicationName`
`CancelledJobs`	Nombre de tâches annulées.	`ApplicationId` , `ApplicationName`

Vous pouvez surveiller les métriques spécifiques au moteur pour les tâches EMR sans serveur exécutées et terminées avec une application spécifique au moteur. UIs Lorsque vous accédez à l'interface utilisateur pour une tâche en cours d'exécution, l'interface utilisateur de l'application en direct s'affiche avec des mises à jour en temps réel. Lorsque vous accédez à l'interface utilisateur pour une tâche terminée, l'interface utilisateur permanente de l'application s'affiche.

Exécution de tâches

Pour vos tâches EMR sans serveur en cours d'exécution, accédez à une interface en temps réel qui fournit des métriques spécifiques au moteur. Vous pouvez utiliser l'interface utilisateur Apache Spark ou l'interface utilisateur Hive Tez pour surveiller et déboguer vos tâches. Pour y accéder UIs, utilisez

la console EMR Studio ou demandez un point de terminaison URL sécurisé avec le. AWS Command Line Interface

Tâches terminées

Pour les tâches EMR sans serveur terminées, utilisez le serveur d'historique Spark ou l'interface utilisateur de Persistent Hive Tez pour accéder aux détails des tâches, aux étapes, aux tâches et aux indicateurs relatifs à l'exécution des tâches Spark ou Hive. Pour y accéder UIs, utilisez la console EMR Studio ou demandez un point de terminaison URL sécurisé avec le. AWS Command Line Interface

Surveillance au niveau du Job Worker

Amazon EMR Serverless envoie à Amazon les métriques suivantes au niveau du travailleur, disponibles dans l'espace de `AWS/EMRServerless` noms et le `Job Worker Metrics` groupe de métriques. CloudWatch EMR Serverless collecte des points de données auprès de travailleurs individuels lors de l'exécution des tâches au niveau de la tâche, du type de travailleur et du niveau. capacity-allocation-type Vous pouvez l'utiliser `ApplicationId` comme dimension pour surveiller plusieurs tâches appartenant à la même application.

> **Note**
>
> Pour afficher le total du processeur et de la mémoire utilisés par une tâche EMR sans serveur lorsque vous consultez les métriques dans la CloudWatch console Amazon, utilisez la statistique sous forme de somme et la période comme 1 minute.

Mesures EMR Serverless au niveau des travailleurs

Métrique	Description	Unit	Dimension	
`WorkerCpu Allocated`	Nombre total de cœurs de vCPU alloués aux travailleurs dans le cadre d'une exécution de tâche.	vCPU	`JobId`, `JobName`, `Applicati onId` , `Applicati onName` , `WorkerType` , et `CapacityA`	

Métrique	Description	Unit	Dimension
			llocation Type
WorkerCpu Used	Nombre total de cœurs de vCPU utilisés par les travailleurs dans le cadre d'une exécution de tâche.	vCPU	JobId, JobName, Applicati onId , Applicati onName , WorkerType , et CapacityA llocation Type
WorkerMem oryAlloca ted	Mémoire totale en Go allouée aux travailleurs dans le cadre d'une exécution de tâche.	Gigaoctets (Go)	JobId, JobName, Applicati onId , Applicati onName , WorkerType , et CapacityA llocation Type
WorkerMem oryUsed	Mémoire totale en Go utilisée par les travaille urs lors de l'exécution d'une tâche.	Gigaoctets (Go)	JobId, JobName, Applicati onId , Applicati onName , WorkerType , et CapacityA llocation Type

Métrique	Description	Unit	Dimension	
`WorkerEphemeralStorageAllocated`	Nombre d'octets de stockage éphémère alloués aux travailleurs dans le cadre d'une exécution de tâche.	Gigaoctets (Go)	`JobId`, `JobName`, `ApplicationId` , `ApplicationName` , `WorkerType` , et `CapacityAllocationType`	
`WorkerEphemeralStorageUsed`	Nombre d'octets de stockage éphémère utilisés par les travailleurs dans le cadre d'une exécution de tâche.	Gigaoctets (Go)	`JobId`, `JobName`, `ApplicationId` , `ApplicationName` , `WorkerType` , et `CapacityAllocationType`	
`WorkerStorageReadBytes`	Nombre d'octets lus depuis le stockage par les travailleurs lors d'une exécution de tâche.	Octets	`JobId`, `JobName`, `ApplicationId` , `ApplicationName` , `WorkerType` , et `CapacityAllocationType`	

Métrique	Description	Unit	Dimension	
`WorkerStorageWriteBytes`	Nombre d'octets écrits dans le stockage par les travailleurs lors d'une exécution de tâche.	Octets	`JobId`, `JobName`, `ApplicationId` , `ApplicationName` , `WorkerType` , et `CapacityAllocationType`	

Les étapes ci-dessous décrivent comment accéder aux différents types de métriques.

Console

Pour accéder à l'interface utilisateur de votre application via la console

1. Accédez à votre application EMR Serverless dans le studio EMR en suivant les instructions de la section Mise en route depuis la console.
2. Pour accéder à une application UIs et à des journaux spécifiques au moteur pour une tâche en cours d'exécution :
   a. Choisissez un poste doté d'un `RUNNING` statut.
   b. Sélectionnez le poste sur la page Détails de la candidature ou accédez à la page des détails du poste correspondant à votre poste.
   c. Dans le menu déroulant Display UI, choisissez Spark UI ou Hive Tez UI pour accéder à l'interface utilisateur de l'application correspondant à votre type de tâche.
   d. Pour accéder aux journaux du moteur Spark, accédez à l'onglet Executors de l'interface utilisateur Spark, puis cliquez sur le lien Logs pour le pilote. Pour accéder aux journaux du moteur Hive, cliquez sur le lien Logs du DAG approprié dans l'interface utilisateur Hive Tez.
3. Pour accéder à une application UIs et à des journaux spécifiques au moteur pour une tâche terminée, procédez comme suit :

a. Choisissez un poste doté d'un `SUCCESS` statut.

b. Sélectionnez le poste sur la page des détails de candidature de votre candidature ou accédez à la page des détails du poste.

c. Dans le menu déroulant de l'interface utilisateur d'affichage, choisissez Spark History Server ou Persistent Hive Tez UI pour accéder à l'interface utilisateur de l'application correspondant à votre type de tâche.

d. Pour accéder aux journaux du moteur Spark, accédez à l'onglet Executors de l'interface utilisateur Spark, puis cliquez sur le lien Logs pour le pilote. Pour accéder aux journaux du moteur Hive, cliquez sur le lien Logs du DAG approprié dans l'interface utilisateur Hive Tez.

AWS CLI

Pour accéder à l'interface utilisateur de votre application à l'aide du AWS CLI

- Pour générer une URL permettant d'accéder à l'interface utilisateur de votre application pour les tâches en cours et terminées, appelez l'`GetDashboardForJobRun`API.

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

L'URL que vous générez est valide pendant une heure.

Surveillez les métriques de Spark avec Amazon Managed Service pour Prometheus

Avec les versions 7.1.0 et supérieures d'Amazon EMR, vous pouvez intégrer EMR Serverless à Amazon Managed Service for Prometheus afin de collecter des métriques Apache Spark pour les tâches et applications EMR Serverless. Cette intégration est disponible lorsque vous soumettez une tâche ou créez une application à l'aide de la AWS console, de l'API EMR Serverless ou du. AWS CLI

Conditions préalables

Avant de pouvoir transmettre vos métriques Spark à Amazon Managed Service for Prometheus, vous devez remplir les conditions préalables suivantes.

- Créez un espace de travail Amazon Managed Service pour Prometheus. Cet espace de travail sert de point de terminaison pour l'ingestion. Prenez note de l'URL affichée pour Endpoint - URL d'écriture à distance. Vous devrez spécifier l'URL lorsque vous créerez votre application EMR Serverless.
- Pour autoriser l'accès à vos tâches à Amazon Managed Service for Prometheus à des fins de surveillance, ajoutez la politique suivante à votre rôle d'exécution des tâches.

```
{
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

Configuration

Pour utiliser la AWS console afin de créer une application intégrée à Amazon Managed Service for Prometheus

1. Consultez Getting started with Amazon EMR Serverless pour créer une application.
2. Lorsque vous créez une application, choisissez Utiliser les paramètres personnalisés, puis configurez votre application en spécifiant les informations dans les champs que vous souhaitez configurer.
3. Sous Journaux et métriques des applications, choisissez Deliver les métriques du moteur à Amazon Managed Service for Prometheus, puis spécifiez votre URL d'écriture à distance.
4. Spécifiez les autres paramètres de configuration souhaités, puis choisissez Create and start application.

Utilisez l'API AWS CLI ou EMR Serverless

Vous pouvez également utiliser l'API AWS CLI EMR Serverless pour intégrer votre application EMR Serverless à Amazon Managed Service for Prometheus lorsque vous exécutez les commandes ou. `create-application` `start-job-run`

create-application

```
aws emr-serverless create-application \
```

```
--release-label emr-7.1.0 \
--type "SPARK" \
--monitoring-configuration '{
    "prometheusMonitoringConfiguration": {
        "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
    }
}'
```

start-job-run

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
        "entryPointArguments": ["10000"],
        "sparkSubmitParameters": "--conf spark.dynamicAllocation.maxExecutors=10"
    }
}' \
--configuration-overrides '{
     "monitoringConfiguration": {
        "prometheusMonitoringConfiguration": {
            "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
        }
    }
}'
```

L'inclusion `prometheusMonitoringConfiguration` dans votre commande indique qu'EMR Serverless doit exécuter la tâche Spark avec un agent qui collecte les métriques Spark et les écrit sur votre point de terminaison `remoteWriteUrl` pour Amazon Managed Service for Prometheus. Vous pouvez ensuite utiliser les métriques Spark dans Amazon Managed Service for Prometheus pour la visualisation, les alertes et les analyses.

Propriétés de configuration avancées

EMR Serverless utilise un composant de Spark nommé `PrometheusServlet` pour collecter les métriques Spark et traduire les données de performance en données compatibles avec Amazon Managed Service for Prometheus. Par défaut, EMR Serverless définit les valeurs par défaut dans

Spark et analyse les métriques du pilote et de l'exécuteur lorsque vous soumettez une tâche à l'aide de. `PrometheusMonitoringConfiguration`

Le tableau suivant décrit toutes les propriétés configurées lors de la soumission d'une tâche Spark qui envoie des métriques à Amazon Managed Service for Prometheus.

Propriété Spark	Valeur par défaut	Description
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	Classe utilisée par Spark pour envoyer des métriques à Amazon Managed Service for Prometheus. Pour modifier le comportement par défaut, spécifiez votre propre classe personnalisée.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source. JvmSource	La classe utilisée par Spark pour collecter et envoyer des métriques cruciales à partir de la machine virtuelle Java sous-jacente. Pour arrêter de collecter des métriques JVM, désactivez cette propriété en lui attribuant une chaîne vide, telle que`""`. Pour modifier le comportement par défaut, spécifiez votre propre classe personnalisée.
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/métriques/prométhée	URL distincte utilisée par Amazon Managed Service for Prometheus pour collecter les statistiques du pilote. Pour modifier le comportement par défaut, spécifiez votre propre chemin. Pour arrêter de collecter les métriques relatives aux conducteurs,

Propriété Spark	Valeur par défaut	Description
		désactivez cette propriété en lui attribuant une chaîne vide, telle que''.
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/executor/prometheus	URL distincte utilisée par Amazon Managed Service for Prometheus pour collecter des métriques auprès de l'exécuteur. Pour modifier le comportement par défaut, spécifiez votre propre chemin. Pour arrêter de collecter les métriques de l'exécuteur, désactivez cette propriété en lui attribuant une chaîne vide, telle que''.

Pour plus d'informations sur les métriques Spark, reportez-vous aux métriques Apache Spark.

Considérations et restrictions

Lorsque vous utilisez Amazon Managed Service for Prometheus pour collecter des métriques à partir d'EMR Serverless, tenez compte des considérations et limites suivantes.

- Support pour l'utilisation d'Amazon Managed Service for Prometheus avec EMR Serverless uniquement dans les pays où Amazon Managed Service for Régions AWS Prometheus est généralement disponible.
- L'exécution de l'agent pour collecter les métriques Spark sur Amazon Managed Service for Prometheus nécessite davantage de ressources de la part des employés. Si vous choisissez une taille de travail plus petite, telle qu'un travailleur vCPU, la durée d'exécution de votre tâche peut augmenter.
- Support pour l'utilisation d'Amazon Managed Service for Prometheus avec EMR Serverless uniquement pour les versions 7.1.0 et supérieures d'Amazon EMR.
- Amazon Managed Service for Prometheus doit être déployé sur le même compte sur lequel vous exécutez EMR Serverless pour collecter des métriques.

Mesures d'utilisation EMR sans serveur

Vous pouvez utiliser les statistiques CloudWatch d'utilisation d'Amazon pour avoir une visibilité sur les ressources utilisées par votre compte. Utilisez ces indicateurs pour visualiser l'utilisation de vos services sur CloudWatch des graphiques et des tableaux de bord.

Les métriques d'utilisation EMR Serverless correspondent aux Service Quotas. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche d'un quota de service. Pour plus d'informations, reportez-vous aux rubriques Service Quotas et Amazon CloudWatch alarmes du Guide de l'utilisateur de Service Quotas.

Pour plus d'informations sur les quotas de service EMR sans serveur, reportez-vous à. Points de terminaison et quotas pour EMR Serverless

Mesures d'utilisation des quotas de service pour EMR Serverless

EMR Serverless publie les mesures d'utilisation des quotas de service suivantes dans l'espace de noms. `AWS/Usage`

Métrique	Description
`ResourceCount`	Le nombre total de ressources spécifiées en cours d'exécution sur votre compte. La ressource est définie par les dimensions associées à la métrique.

Dimensions des métriques d'utilisation des quotas de service EMR Serverless

Vous pouvez utiliser les dimensions suivantes pour affiner les métriques d'utilisation publiées par EMR Serverless.

Dimension	Value	Description
`Service`	EMR sans serveur	Le nom de la ressource Service AWS qui contient la ressource.

Dimension	Value	Description
`Type`	Ressource	Type d'entité signalée par EMR Serverless.
`Resource`	vCPU	Type de ressource suivie par EMR Serverless.
`Class`	Aucune	Classe de ressource suivie par EMR Serverless.

Automatiser l'EMR sans serveur avec Amazon EventBridge

Vous pouvez l'utiliser Amazon EventBridge pour automatiser Services AWS et répondre automatiquement aux événements du système, tels que les problèmes de disponibilité des applications ou les modifications des ressources. EventBridge fournit un flux d'événements système en temps quasi réel décrivant les modifications apportées à vos AWS ressources. Vous pouvez écrire des règles simples pour indiquer quels événements vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Avec EventBridge, vous pouvez automatiquement :

- Invoquer une AWS Lambda fonction
- Transmettre un événement à Amazon Kinesis Data Streams
- Activer une machine à AWS Step Functions états
- Notifier une rubrique Amazon SNS ou une file d'attente Amazon SQS

Par exemple, lorsque vous utilisez EventBridge EMR Serverless, vous pouvez activer une AWS Lambda fonction en cas de réussite d'une tâche ETL ou informer une rubrique Amazon SNS en cas d'échec d'une tâche ETL.

EMR Serverless émet quatre types d'événements :

- Événements de modification de l'état d'une application : événements qui déclenchent chaque changement d'état d'une application. Pour plus d'informations sur les états des applications, reportez-vous àÉtats de l'application.

- Événements de changement d'état d'exécution d'une tâche : événements qui émettent chaque changement d'état d'une exécution de tâche. Pour plus d'informations sur, reportez-vous àÉtats d'exécution de la tâche.
- Événements de nouvelle tentative d'exécution de tâches : événements qui émettent chaque nouvelle tentative d'exécution d'une tâche à partir des versions 7.1.0 et supérieures d'Amazon EMR Serverless.
- Événements de mise à jour de l'utilisation des ressources d'une tâche : événements qui émettent des mises à jour sur l'utilisation des ressources pour une tâche exécutée à des intervalles de près de 30 minutes.

Exemples d'événements EMR sans serveur EventBridge

Les événements signalés par EMR Serverless ont la valeur `aws.emr-serverless` attribuée à`source`, comme dans les exemples suivants.

Événement de changement d'état de l'application

L'exemple d'événement suivant montre une application dans son `CREATING` état.

```
{
    "version": "0",
    "id": "9fd3cf79-1ff1-b633-4dd9-34508dc1e660",
    "detail-type": "EMR Serverless Application State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:16:31Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "applicationId": "00f1cbsc6anuij25",
        "applicationName": "3965ad00-8fba-4932-a6c8-ded32786fd42",
        "arn": "arn:aws:emr-serverless:us-east-1:111122223333:/
applications/00f1cbsc6anuij25",
        "releaseLabel": "emr-6.6.0",
        "state": "CREATING",
        "type": "HIVE",
        "createdAt": "2022-05-31T21:16:31.547953Z",
        "updatedAt": "2022-05-31T21:16:31.547970Z",
        "autoStopConfig": {
            "enabled": true,
```

```
            "idleTimeout": 15
        },
        "autoStartConfig": {
            "enabled": true
        }
    }
}
```

Événement de changement d'état de Job Run

L'exemple d'événement suivant montre une exécution de travail qui passe de l'`SCHEDULED`état à l'`RUNNING`état.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "state": "RUNNING",
        "previousState": "SCHEDULED",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z"
    }
}
```

Événement Job Run Retry

Voici un exemple d'événement de nouvelle tentative d'exécution d'une tâche.

```
{
    "version": "0",
```

```
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run Retry",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        //Attempt Details
        "previousAttempt": 1,
        "previousAttemptState": "FAILED",
        "previousAttemptCreatedAt": "2022-05-31T21:07:25.325900Z",
        "previousAttemptEndedAt": "2022-05-31T21:07:30.325900Z",
        "newAttempt": 2,
        "newAttemptCreatedAt": "2022-05-31T21:07:30.325900Z"
    }
}
```

Mise à jour sur l'utilisation des ressources de travail

L'exemple d'événement suivant montre la dernière mise à jour de l'utilisation des ressources pour une tâche passée à l'état terminal après son exécution.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Resource Utilization Update",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:emr-serverless:us-east-1:123456789012:/applications/00f1982r1uukb925/
jobruns/00f1cbn5g4bb0c01"
```

```
    ],
    "detail": {
        "applicationId": "00f1982r1uukb925",
        "jobRunId": "00f1cbn5g4bb0c01",
        "attempt": 1,
        "mode": "BATCH",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        "startedAt": "2022-05-31T21:07:26.123Z",
        "calculatedFrom": "2022-05-31T21:07:42.299487Z",
        "calculatedTo": "2022-05-31T21:07:30.325900Z",
        "resourceUtilizationFinal": true,
        "resourceUtilizationForInterval": {
            "vCPUHour": 0.023,
            "memoryGBHour": 0.114,
            "storageGBHour": 0.228
        },
        "billedResourceUtilizationForInterval": {
            "vCPUHour": 0.067,
            "memoryGBHour": 0.333,
            "storageGBHour": 0
        },
        "totalResourceUtilization": {
            "vCPUHour": 0.023,
            "memoryGBHour": 0.114,
            "storageGBHour": 0.228
        },
        "totalBilledResourceUtilization": {
            "vCPUHour": 0.067,
            "memoryGBHour": 0.333,
            "storageGBHour": 0
        }
    }
}
```

Le champ StarteDat ne sera présent que si la tâche est passée à l'état en cours d'exécution.

Balisage des ressources

Attribuez vos propres métadonnées à chaque ressource à l'aide de balises pour vous aider à gérer vos ressources EMR Serverless. Cette section fournit une vue d'ensemble des fonctions des balises et explique comment créer des balises.

Rubriques

Qu'est-ce qu'un tag ?

Une étiquette est une étiquette que vous attribuez à une AWS ressource. Chaque balise est constituée d'une clé et d'une valeur que vous définissez toutes deux. Les balises vous permettent de classer vos AWS ressources en fonction d'attributs tels que l'objectif, le propriétaire et l'environnement. Lorsque vous disposez de nombreuses ressources du même type, identifiez rapidement une ressource spécifique en fonction des balises qui lui sont attribuées. Par exemple, définissez un ensemble de balises pour vos applications Amazon EMR Serverless afin de suivre le propriétaire et le niveau de pile de chaque application. Nous vous suggérons de concevoir un ensemble cohérent de clés de balise pour chaque type de ressource.

Les balises ne sont pas automatiquement affectées à vos ressources. Après avoir ajouté une balise à une ressource, modifiez la valeur d'une balise ou supprimez-la de la ressource à tout moment. Les balises n'ont aucune signification sémantique pour Amazon EMR Serverless et sont interprétées uniquement comme des chaînes de caractères. 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 IAM, vous pouvez contrôler quels utilisateurs de votre AWS compte sont autorisés à gérer les balises. Pour des exemples de politiques de contrôle d'accès basées sur des balises, reportez-vous àPolitiques de contrôle d'accès basées sur les balises.

Balisage de vos ressources

Vous pouvez étiqueter des applications et des cycles de travail nouveaux ou existants. Si vous utilisez l'API Amazon EMR Serverless, le ou un AWS SDK AWS CLI, vous pouvez appliquer des balises aux nouvelles ressources à l'aide du `tags` paramètre 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 balisez des ressources au moment de leur création, il n'est pas nécessaire d'exécuter des scripts de balisage personnalisés après avoir créé une ressource.

Le tableau suivant décrit les ressources Amazon EMR Serverless qui peuvent être balisées.

Ressources pouvant être balisées

Ressource	Prend en charge les étiquettes	Prend en charge la propagation des étiquettes	Prend en charge le balisage lors de la création (API AWS CLI Amazon EMR Serverless et SDK) AWS	API de création (des balises peuvent être ajoutées lors de la création)
Application	Oui	Non Les balises associées à une application ne se propagent pas aux exécution s de tâches soumises à cette application.	Oui	`CreateApp lication`

Ressource	Prend en charge les étiquettes	Prend en charge la propagation des étiquettes	Prend en charge le balisage lors de la création (API AWS CLI Amazon EMR Serverless et SDK) AWS	API de création (des balises peuvent être ajoutées lors de la création)
Exécution de tâche	Oui	Non	Oui	`StartJobRun`

Limites relatives au balisage

Les restrictions de base suivantes s'appliquent aux balises :

- Chaque ressource peut avoir un maximum de 50 balises créées par l'utilisateur.
- Pour chaque ressource, chaque clé d'identification doit être unique, et chaque clé d'identification peut avoir une seule valeur.
- La longueur maximale de la clé est de 128 caractères Unicode en UTF-8.
- La longueur maximale de la valeur est de 256 caractères Unicode en UTF-8.
- Les caractères autorisés sont les lettres, les chiffres, les espaces représentables en UTF-8 et les caractères suivants : _. :/= + - @.
- Une clé de balise ne peut pas être une chaîne vide. Une valeur de balise peut être une chaîne vide, mais pas null.
- Les clés et les valeurs des balises distinguent les majuscules et minuscules.
- N'utilisez `AWS:` pas de combinaison majuscules ou minuscules, telle qu'un préfixe pour les clés ou les valeurs. Celles-ci ne peuvent être utilisées que pour AWS .

Utilisation des balises à l'aide de l'API sans serveur Amazon EMR AWS CLI et de l'API Amazon EMR

Utilisez les AWS CLI commandes ou les opérations d'API Amazon EMR Serverless suivantes pour ajouter, mettre à jour, répertorier et supprimer les balises de vos ressources.

Commandes CLI et opérations d'API pour les balises

Ressource	Prend en charge les étiquettes	Prend en charge la propagation des étiquettes
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 baliser ou débaliser des ressources à l'aide du AWS CLI.

Marquer une application existante

La commande suivante balise une application existante.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

Annuler le balisage d'une application existante

La commande suivante supprime une balise d'une application existante.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Répertorier les balises d'une ressource

La commande suivante permet de répertorier l'ensemble des étiquettes associées à une ressource existante.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

Tutoriels pour EMR Serverless

Cette section décrit les cas d'utilisation courants lorsque vous travaillez avec des applications EMR sans serveur. Cela inclut une variété d'outils, notamment Hudi et Iceberg, pour travailler sur de grands ensembles de données et utiliser Python et les bibliothèques Python pour soumettre des tâches Spark.

Rubriques

Utilisation de Java 17 avec Amazon EMR Serverless

Avec les versions 6.11.0 et supérieures d'Amazon EMR, configurez les tâches EMR Serverless Spark pour utiliser le runtime Java 17 pour la machine virtuelle Java (JVM). Utilisez l'une des méthodes suivantes pour configurer Spark avec Java 17.

`JAVA_HOME`

Pour remplacer le paramètre JVM pour EMR Serverless 6.11.0 et versions ultérieures, fournissez le paramètre à sa classification et à sa classification d'environnement. `JAVA_HOME` `spark.emr-serverless.driverEnv` `spark.executorEnv`

x86_64

Définissez les propriétés requises pour spécifier Java 17 comme `JAVA_HOME` configuration pour le pilote et les exécuteurs Spark :

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.x86_64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64/
```

arm_64

Définissez les propriétés requises pour spécifier Java 17 comme `JAVA_HOME` configuration pour le pilote et les exécuteurs Spark :

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.aarch64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.aarch64/
```

`spark-defaults`

Vous pouvez également spécifier Java 17 dans la `spark-defaults` classification pour remplacer le paramètre JVM pour EMR Serverless 6.11.0 et versions ultérieures.

x86_64

Spécifiez Java 17 dans la `spark-defaults` classification :

```
{
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.x86_64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.x86_64/"
            }
         }
      ]
}
```

arm_64

Spécifiez Java 17 dans la `spark-defaults` classification :

```
{
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.aarch64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.aarch64/"
            }
         }
      ]
}
```

Utilisation d'Apache Hudi avec EMR sans serveur

Cette section décrit l'utilisation d'Apache Hudi avec des applications EMR sans serveur. Hudi est un framework de gestion des données qui simplifie le traitement des données.

Pour utiliser Apache Hudi avec des applications EMR sans serveur

1. Définissez les propriétés Spark requises lors de l'exécution de la tâche Spark correspondante.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,/usr/lib/hudi/hudi-utilities-
   bundle.jar,/usr/lib/hudi/hudi-aws-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Pour synchroniser une table Hudi avec le catalogue configuré, désignez le catalogue de données AWS Glue comme métastore ou configurez un métastore externe. EMR Serverless est compatible avec le mode de synchronisation hms des tables Hive pour les charges de travail Hudi. EMR Serverless active cette propriété par défaut. Pour en savoir plus sur la configuration de votre métastore, reportez-vous à. Configuration du métastore pour EMR Serverless

> **Important**
>
> EMR Serverless ne prend pas en charge `HIVEQL` ou n'est pas disponible en `JDBC` tant qu'option de mode de synchronisation pour les tables Hive afin de gérer les charges de travail Hudi. Pour en savoir plus, reportez-vous à la section Modes de synchronisation.

Lorsque vous utilisez le catalogue de données AWS Glue comme métastore, spécifiez les propriétés de configuration suivantes pour votre tâche Hudi.

```
--conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer,
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Pour en savoir plus sur les versions Apache Hudi d'Amazon EMR, consultez l'historique des versions de Hudi.

Utilisation d'Apache Iceberg avec EMR sans serveur

Cette section décrit comment utiliser Apache Iceberg avec des applications EMR sans serveur. Apache Iceberg est un format de table qui permet de travailler avec de grands ensembles de données dans des lacs de données.

Pour utiliser Apache Iceberg avec des applications EMR sans serveur

1. Définissez les propriétés Spark requises lors de l'exécution de la tâche Spark correspondante.

   ```
   spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
   ```

2. Désignez le AWS Glue Data Catalog comme métastore ou configurez un métastore externe. Pour en savoir plus sur la configuration de votre métastore, reportez-vous à. Configuration du métastore pour EMR Serverless

   Configurez les propriétés du métastore que vous souhaitez utiliser pour Iceberg. Par exemple, si vous souhaitez utiliser le catalogue de données AWS Glue, définissez les propriétés suivantes dans la configuration de l'application.

```
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.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
```

Lorsque vous utilisez le catalogue de données AWS Glue comme métastore, spécifiez les propriétés de configuration suivantes pour votre tâche Iceberg.

```
--conf spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar,
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,
--conf spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog,
--conf spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog,
--conf spark.sql.catalog.dev.warehouse=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Pour en savoir plus sur les versions d'Apache Iceberg d'Amazon EMR, consultez l'historique des versions d'Iceberg.

Utilisation de bibliothèques Python avec EMR Serverless

Lorsque vous exécutez PySpark des tâches sur des applications Amazon EMR Serverless, empaquetez différentes bibliothèques Python en tant que dépendances. Pour ce faire, utilisez les fonctionnalités natives de Python, créez un environnement virtuel ou configurez directement vos PySpark tâches pour utiliser les bibliothèques Python. Cette page couvre chaque approche.

Utilisation des fonctionnalités natives de Python

Lorsque vous définissez la configuration suivante, utilisez-la PySpark pour télécharger des fichiers Python (`.py`), des packages Python compressés (`.zip`) et des fichiers Egg (`.egg`) vers les exécuteurs Spark.

```
--conf spark.submit.pyFiles=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
 file>
```

Pour plus de détails sur l'utilisation des environnements virtuels Python pour les PySpark tâches, reportez-vous à la section Utilisation des fonctionnalités PySpark natives.

Lorsque vous utilisez EMR Notebook, vous pouvez rendre la dépendance Python disponible dans votre bloc-notes en exécutant le code suivant :

```
   %%configure -f
{
   "conf": {
   "spark.submit.pyFiles":"s3:///amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
file>
                  }
}
```

Création d'un environnement virtuel Python

Pour empaqueter plusieurs bibliothèques Python pour une PySpark tâche, créez des environnements virtuels Python isolés.

1. Pour créer l'environnement virtuel Python, utilisez les commandes suivantes. L'exemple illustré installe les packages `scipy` `matplotlib` dans un environnement virtuel et copie l'archive vers un emplacement Amazon S3.

   > **Important**
   >
   > Vous devez exécuter les commandes suivantes dans un environnement Amazon Linux 2 similaire avec la même version de Python que celle que vous utilisez dans EMR Serverless, c'est-à-dire Python 3.7.10 pour Amazon EMR version 6.6.0. Vous trouverez un exemple de Dockerfile dans le référentiel EMR Serverless Samples. GitHub

   ```
   # initialize a python virtual environment
   python3 -m venv pyspark_venvsource
   source pyspark_venvsource/bin/activate

   # optionally, ensure pip is up-to-date
   pip3 install --upgrade pip

   # install the python packages
   pip3 install scipy
   ```

```
pip3 install matplotlib

# package the virtual environment into an archive
pip3 install venv-pack
venv-pack -f -o pyspark_venv.tar.gz

# copy the archive to an S3 location
aws s3 cp pyspark_venv.tar.gz s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Soumettez la tâche Spark avec vos propriétés définies pour utiliser l'environnement virtuel Python.

```
--conf spark.archives=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Notez que si vous ne remplacez pas le binaire Python d'origine, la deuxième configuration de la séquence de paramètres précédente sera `--conf spark.executorEnv.PYSPARK_PYTHON=python` la suivante.

Pour en savoir plus sur l'utilisation des environnements virtuels Python pour les PySpark tâches, reportez-vous à la section Utilisation de Virtualenv. Pour plus d'exemples sur la façon de soumettre des tâches Spark, reportez-vous àUtilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur.

Configuration des PySpark tâches pour utiliser les bibliothèques Python

Avec les versions 6.12.0 et supérieures d'Amazon EMR, vous pouvez configurer directement les tâches EMR Serverless PySpark pour utiliser les bibliothèques Python de science des données les plus populaires, telles que Pandas, et ce, sans aucune configuration supplémentaire. NumPyPyArrow

Les exemples suivants montrent comment empaqueter chaque bibliothèque Python pour une PySpark tâche.

NumPy

NumPy est une bibliothèque Python pour le calcul scientifique qui propose des tableaux multidimensionnels et des opérations pour les mathématiques, le tri, la simulation aléatoire et les statistiques de base. Pour l'utiliser NumPy, exécutez la commande suivante :

```
import numpy
```

pandas

pandas est une bibliothèque Python construite sur NumPy. La bibliothèque Pandas fournit aux scientifiques des structures de DataFramedonnées et des outils d'analyse de données. Pour utiliser les pandas, exécutez la commande suivante :

```
import pandas
```

PyArrow

PyArrow est une bibliothèque Python qui gère les données colonnaires en mémoire pour améliorer les performances au travail. PyArrow est basé sur la spécification de développement multilingue Apache Arrow, qui est un moyen standard de représenter et d'échanger des données dans un format en colonnes. Pour l'utiliser PyArrow, exécutez la commande suivante :

```
import pyarrow
```

Utilisation de différentes versions de Python avec EMR Serverless

Outre le cas d'utilisationUtilisation de bibliothèques Python avec EMR Serverless, vous pouvez également utiliser des environnements virtuels Python pour travailler avec des versions de Python différentes de celles incluses dans la version Amazon EMR pour votre application Amazon EMR Serverless. Pour ce faire, créez un environnement virtuel Python avec la version de Python que vous souhaitez utiliser.

Pour soumettre une tâche depuis un environnement virtuel Python

1. Créez votre environnement virtuel à l'aide des commandes de l'exemple suivant. Cet exemple installe Python 3.9.9 dans un package d'environnement virtuel et copie l'archive vers un emplacement Amazon S3.

> **Important**
>
> Si vous utilisez les versions 7.0.0 et supérieures d'Amazon EMR, exécutez vos commandes dans un environnement Amazon Linux 2023 similaire à celui que vous utilisez pour vos applications EMR Serverless.
> Si vous utilisez la version 6.15.0 ou une version antérieure, exécutez les commandes suivantes dans un environnement Amazon Linux 2 similaire.

```
# install Python 3.9.9 and activate the venv
yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz && \
tar xzf Python-3.9.9.tgz && cd Python-3.9.9 && \
./configure --enable-optimizations && \
make altinstall

# create python venv with Python 3.9.9
python3.9 -m venv pyspark_venv_python_3.9.9 --copies
source pyspark_venv_python_3.9.9/bin/activate

# copy system python3 libraries to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/

# package venv to archive.
# **Note** that you have to supply --python-prefix option
# to make sure python starts with the path where your
# copied libraries are present.
# Copying the python binary to the "environment" directory.
pip3 install venv-pack
venv-pack -f -o pyspark_venv_python_3.9.9.tar.gz --python-prefix /home/hadoop/
environment

# stage the archive in S3
aws s3 cp pyspark_venv_python_3.9.9.tar.gz s3://<path>

# optionally, remove the virtual environment directory
rm -fr pyspark_venv_python_3.9.9
```

2. Définissez vos propriétés pour utiliser l'environnement virtuel Python et soumettez le job Spark.

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Pour en savoir plus sur l'utilisation des environnements virtuels Python pour les PySpark tâches, reportez-vous à la section Utilisation de Virtualenv. Pour plus d'exemples sur la façon de soumettre des tâches Spark, reportez-vous àUtilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur.

Utilisation de Delta Lake OSS avec EMR Serverless

Amazon EMR versions 6.9.0 et supérieures

> **Note**
>
> Amazon EMR 7.0.0 et versions ultérieures utilisent Delta Lake 3.0.0, qui renomme le fichier en. `delta-core.jar` `delta-spark.jar` Si vous utilisez Amazon EMR 7.0.0 ou une version ultérieure, assurez-vous de le spécifier `delta-spark.jar` dans vos configurations.

Amazon EMR 6.9.0 et versions ultérieures inclut Delta Lake. Vous n'avez donc plus à emballer Delta Lake vous-même ni à fournir le `--packages` drapeau avec vos tâches EMR sans serveur.

1. Lorsque vous soumettez des tâches EMR sans serveur, assurez-vous de disposer des propriétés de configuration suivantes et d'inclure les paramètres suivants dans le champ. `sparkSubmitParameters`

```
--conf spark.jars=/usr/share/aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/
delta-storage.jar
    --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension
    --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
```

2. Créez un local `delta_sample.py` pour tester la création et la lecture d'une table Delta.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://amzn-s3-demo-bucket/delta-lake/output/%s/" % str(uuid.uuid4())
    spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

    ## creates a Delta table and outputs to target S3 bucket
    spark.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    spark.read.format("delta").load(url).show
```

3. À l'aide de AWS CLI, téléchargez le `delta_sample.py` fichier dans votre compartiment Amazon S3. Utilisez ensuite la `start-job-run` commande pour soumettre une tâche à une application EMR sans serveur existante.

```
aws s3 cp delta_sample.py s3://amzn-s3-demo-bucket/code/

    aws emr-serverless start-job-run \
        --application-id application-id \
        --execution-role-arn job-role-arn \
        --name emr-delta \
        --job-driver '{
            "sparkSubmit": {
                "entryPoint": "s3://amzn-s3-demo-bucket/code/delta_sample.py",
                "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
            }
        }'
```

Pour utiliser les bibliothèques Python avec Delta Lake, ajoutez la `delta-core` bibliothèque en l'empaquetant sous forme de dépendance ou en l'utilisant comme image personnalisée.

Vous pouvez également utiliser le `SparkContext.addPyFile` pour ajouter les bibliothèques Python à partir du fichier `delta-core` JAR :

```
import glob
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
spark.sparkContext.addPyFile(glob.glob("/usr/share/aws/delta/lib/delta-core_*.jar")[0])
```

Amazon EMR versions 6.8.0 et antérieures

Si vous utilisez Amazon EMR 6.8.0 ou une version antérieure, suivez ces étapes pour utiliser Delta Lake OSS avec vos applications EMR Serverless.

1. Pour créer une version open source de Delta Lake compatible avec la version de Spark sur votre application Amazon EMR Serverless, accédez au Delta GitHub et suivez les instructions.
2. Téléchargez les bibliothèques Delta Lake dans un compartiment Amazon S3 de votre Compte AWS.
3. Lorsque vous soumettez des tâches EMR sans serveur dans la configuration de l'application, incluez les fichiers JAR de Delta Lake qui se trouvent désormais dans votre compartiment.

   ```
   --conf spark.jars=s3://amzn-s3-demo-bucket/jars/delta-core_2.12-1.1.0.jar
   ```

4. Pour vous assurer que vous pouvez lire et écrire à partir d'une table Delta, exécutez un exemple de PySpark test.

   ```
   from pyspark import SparkConf, SparkContext
       from pyspark.sql import HiveContext, SparkSession

       import uuid

       conf = SparkConf()
       sc = SparkContext(conf=conf)
       sqlContext = HiveContext(sc)

       url = "s3://amzn-s3-demo-bucket/delta-lake/output/1.0.1/%s/" %
    str(uuid.uuid4())

       ## creates a Delta table and outputs to target S3 bucket
       session.range(5).write.format("delta").save(url)

       ## reads a Delta table and outputs to target S3 bucket
       session.read.format("delta").load(url).show
   ```

Soumission de tâches EMR sans serveur depuis Airflow

Le fournisseur Amazon d'Apache Airflow fournit des opérateurs EMR sans serveur. Pour plus d'informations sur les opérateurs, consultez la section Amazon EMR Serverless Operators dans la documentation Apache Airflow.

Vous pouvez l'utiliser `EmrServerlessCreateApplicationOperator` pour créer une application Spark ou Hive. Vous pouvez également l'utiliser `EmrServerlessStartJobOperator` pour démarrer une ou plusieurs tâches avec votre nouvelle application.

Pour utiliser l'opérateur avec Amazon Managed Workflows for Apache Airflow (MWAA) avec Airflow 2.2.2, ajoutez la ligne suivante à votre `requirements.txt` fichier et mettez à jour votre environnement MWAA pour utiliser le nouveau fichier.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Notez que le support EMR Serverless a été ajouté à la version 5.0.0 du fournisseur Amazon. La version 6.0.0 est la dernière version compatible avec Airflow 2.2.2. Vous pouvez utiliser les versions ultérieures avec Airflow 2.4.3 sur MWAA.

L'exemple abrégé suivant montre comment créer une application, exécuter plusieurs tâches Spark, puis arrêter l'application. Un exemple complet est disponible dans le référentiel EMR Serverless Samples. GitHub Pour plus de détails sur `sparkSubmit` la configuration, reportez-vous àUtilisation des configurations Spark lorsque vous exécutez des tâches EMR sans serveur.

```
from datetime import datetime

from airflow import DAG
from airflow.providers.amazon.aws.operators.emr import (
    EmrServerlessCreateApplicationOperator,
    EmrServerlessStartJobOperator,
    EmrServerlessDeleteApplicationOperator,
)

# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "amzn-s3-demo-bucket"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
```

```
        "s3MonitoringConfiguration": {"logUri": f"s3://amzn-s3-demo-bucket/logs/"}
    },
}

with DAG(
    dag_id="example_endtoend_emr_serverless_job",
    schedule_interval=None,
    start_date=datetime(2021, 1, 1),
    tags=["example"],
    catchup=False,
) as dag:
    create_app = EmrServerlessCreateApplicationOperator(
        task_id="create_spark_app",
        job_type="SPARK",
        release_label="emr-6.7.0",
        config={"name": "airflow-test"},
    )

    application_id = create_app.output

    job1 = EmrServerlessStartJobOperator(
        task_id="start_job_1",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    job2 = EmrServerlessStartJobOperator(
        task_id="start_job_2",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
                "entryPointArguments": ["1000"]
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
```

```
    )

    delete_app = EmrServerlessDeleteApplicationOperator(
        task_id="delete_app",
        application_id=application_id,
        trigger_rule="all_done",
    )

    (create_app >> [job1, job2] >> delete_app)
```

Utilisation des fonctions définies par l'utilisateur de Hive avec EMR Serverless

Les fonctions définies par l'utilisateur Hive (UDFs) vous permettent de créer des fonctions personnalisées pour traiter des enregistrements ou des groupes d'enregistrements. Dans ce didacticiel, vous allez utiliser un exemple d'UDF avec une application Amazon EMR Serverless préexistante pour exécuter une tâche qui génère un résultat de requête. Pour savoir comment configurer une application, reportez-vous àCommencer à utiliser Amazon EMR Serverless.

Pour utiliser un UDF avec EMR Serverless

1. Accédez au GitHubpour obtenir un exemple d'UDF. Clonez le dépôt et passez à la branche git que vous souhaitez utiliser. Mettez à jour le `maven-compiler-plugin` dans le `pom.xml` fichier du référentiel pour avoir une source. Mettez également à jour la configuration de la version Java cible vers`1.8`. Exécutez `mvn package -DskipTests` pour créer le fichier JAR contenant votre exemple UDFs.
2. Après avoir créé le fichier JAR, chargez-le dans votre compartiment S3 à l'aide de la commande suivante.

   ```
   aws s3 cp brickhouse-0.8.2-JS.jar s3://amzn-s3-demo-bucket/jars/
   ```

3. Créez un fichier d'exemple pour utiliser l'une des fonctions UDF d'exemple. Enregistrez cette requête sous `udf_example.q` et chargez-la dans votre compartiment S3.

   ```
   add jar s3://amzn-s3-demo-bucket/jars/brickhouse-0.8.2-JS.jar;
   CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
   select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
    array(cast(0 as int))));
   ```

```
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))))["key1"][2];
```

4. Soumettez le job Hive suivant.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://amzn-s3-demo-bucket/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/
emr-serverless-hive/warehouse"
    }
}' --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.driver.cores": "2",
            "hive.driver.memory": "6G"
        }
    }],
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/logs/"
        }
    }
}'
```

5. Utilisez la `get-job-run` commande pour vérifier l'état de votre tâche. Attendez que l'état passe à`SUCCESS`.

```
aws emr-serverless get-job-run --application-id application-id --job-run-id job-id
```

6. Téléchargez les fichiers de sortie à l'aide de la commande suivante.

```
aws s3 cp --recursive s3://amzn-s3-demo-bucket/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

Le `stdout.gz` fichier ressemble au suivant.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

Utilisation d'images personnalisées avec EMR Serverless

Rubriques

Utiliser une version personnalisée de Python

Vous pouvez créer une image personnalisée pour utiliser une version différente de Python. Pour utiliser la version 3.10 de Python pour les tâches Spark, par exemple, exécutez la commande suivante :

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install python 3
RUN yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
RUN wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tgz && \
tar xzf Python-3.10.0.tgz && cd Python-3.10.0 && \
./configure --enable-optimizations && \
make altinstall

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

Avant de soumettre le job Spark, définissez vos propriétés pour utiliser l'environnement virtuel Python, comme suit.

```
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=/usr/local/bin/python3.10
```

```
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
--conf spark.executorEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
```

Utiliser une version personnalisée de Java

L'exemple suivant montre comment créer une image personnalisée pour utiliser Java 11 pour vos tâches Spark.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install JDK 11
RUN amazon-linux-extras install java-openjdk11

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

Avant de soumettre la tâche Spark, définissez les propriétés de Spark pour utiliser Java 11, comme suit.

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-1.amzn2.0.1.x86_64
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-
```

Créez une image de science des données

L'exemple suivant montre comment inclure des packages Python courants pour la science des données, tels que Pandas et NumPy.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# python packages
RUN pip3 install boto3 pandas numpy
RUN pip3 install -U scikit-learn==0.23.2 scipy
RUN pip3 install sk-dist
RUN pip3 install xgboost
```

```
# EMR Serverless runs the image as hadoop
USER hadoop:hadoop
```

Traitement des données géospatiales avec Apache Sedona

L'exemple suivant montre comment créer une image pour inclure Apache Sedona pour le traitement géospatial.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

RUN yum install -y wget
RUN wget https://repo1.maven.org/maven2/org/apache/sedona/sedona-core-3.0_2.12/1.3.0-
incubating/sedona-core-3.0_2.12-1.3.0-incubating.jar -P /usr/lib/spark/jars/
RUN pip3 install apache-sedona

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

Informations de licence pour l'utilisation d'images personnalisées

Vous pouvez créer des images personnalisées avec EMR Serverless pour effectuer des tâches spécifiques ou pour utiliser des versions spécifiques d'un progiciel. La modification et la distribution d'images personnalisées peuvent être soumises à des règles et à des conditions de licence. Le texte de licence apparaît dans la sous-section suivante.

Licence applicable aux images personnalisées

Copyright Amazon.com et ses filiales ; tous droits réservés. Ce logiciel est un AWS contenu AWS régi par le contrat client et ne peut être distribué sans autorisation. Outre les autorisations prévues dans la licence de propriétéAWS intellectuelle, le AWS concédant vous accorde les autorisations supplémentaires suivantes :

La création, la copie et l'utilisation de dérivés du AWS contenu sont autorisées à condition que les conditions suivantes soient remplies :

- Vous ne modifiez pas le AWS contenu lui-même, et tout dérivé est strictement le résultat de votre ajout de nouveau contenu.
- Les reproductions internes doivent conserver la mention de copyright ci-dessus.

- La distribution externe, sous forme source ou binaire, avec ou sans modification, n'est pas autorisée selon les termes de cette licence.

Pour plus d'informations sur l'utilisation d'images personnalisées, reportez-vous à la section Utilisation d'images personnalisées avec EMR Serverless.

Utilisation de l'intégration Amazon Redshift pour Apache Spark sur Amazon EMR Serverless

À partir de la version 6.9.0 d'Amazon EMR, chaque image de version inclut un connecteur entre Apache Spark et Amazon Redshift. Avec ce connecteur, utilisez Spark sur Amazon EMR Serverless pour traiter les données stockées dans Amazon Redshift. L'intégration est basée sur le connecteur open-source `spark-redshift`. Pour Amazon EMR Serverless, l'intégration Amazon Redshift pour Apache Spark est incluse en tant qu'intégration native.

Rubriques

- Lancement d'une application Spark avec 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 avec l'intégration Amazon Redshift pour Apache Spark

Pour utiliser l'intégration avec EMR Serverless 6.9.0, transmettez les dépendances Spark-Redshift requises à votre tâche Spark. `--jars`À utiliser pour inclure les bibliothèques associées au connecteur Redshift. Pour accéder aux autres emplacements de fichiers pris en charge par `--jars` cette option, reportez-vous à la section Gestion avancée des dépendances de la documentation d'Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Les versions 6.10.0 et supérieures d'Amazon EMR ne nécessitent pas la dépendance de `minimal-json.jar` et installent automatiquement les autres dépendances sur chaque cluster par défaut. Les exemples suivants montrent comment lancer une application Spark avec l'intégration Amazon Redshift pour Apache Spark.

Amazon EMR 6.10.0 +

Lancez une tâche Spark sur Amazon EMR Serverless avec l'intégration Amazon Redshift pour Apache Spark sur EMR Serverless version 6.10.0 ou ultérieure.

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

Pour lancer une tâche Spark sur Amazon EMR Serverless avec l'intégration Amazon Redshift pour Apache Spark on EMR Serverless version 6.9.0, utilisez l'option illustrée dans l'exemple suivant. `--jars` Notez que les chemins répertoriés avec l'option `--jars` sont les chemins par défaut des fichiers JAR.

```
--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
```

```
spark-submit \
  --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 \
  my_script.py
```

Authentification avec l'intégration Amazon Redshift pour Apache Spark

AWS Secrets Manager À utiliser pour récupérer les informations d'identification et se connecter à Amazon Redshift

Vous pouvez vous authentifier auprès d'Amazon Redshift en toute sécurité en stockant les informations d'identification dans Secrets Manager et en demandant à la tâche Spark d'appeler `GetSecretValue` l'API pour les récupérer :

```
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
```

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

Vous pouvez authentifier une tâche Spark auprès d'un cluster Amazon Redshift en spécifiant 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 un problème pour votre application, utilisez 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
```

Utiliser l'authentification basée sur IAM avec le rôle d'exécution de tâches sans serveur Amazon EMR

À partir de la version 6.9.0 d'Amazon EMR Serverless, le pilote Amazon Redshift JDBC 2.1 ou supérieur 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.

Spécifiez plutôt le `jdbc:redshift:iam://` schéma. Cela commande au pilote JDBC d'utiliser votre rôle d'exécution de tâche EMR Serverless pour récupérer automatiquement les informations d'identification. Reportez-vous à la section Configurer une connexion JDBC ou ODBC pour utiliser les informations d'identification IAM dans le guide de gestion Amazon Redshift pour plus d'informations. Voici un exemple de cette URL :

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Les autorisations suivantes sont requises pour votre rôle d'exécution de tâches lorsque les conditions fournies sont remplies :

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`	Obligatoire si vous utilisez Amazon Redshift Serverless et que vous spécifiez l'URL en termes de nom du groupe de travail et de région

Connexion à Amazon Redshift au sein d'un autre VPC

Lorsque vous configurez un cluster Amazon Redshift provisionné ou un groupe de travail Amazon Redshift Serverless dans le cadre d'un VPC, configurez la connectivité VPC pour que votre

application Amazon EMR Serverless puisse accéder aux ressources. Pour plus d'informations sur la configuration de la connectivité VPC sur une application EMR sans serveur, reportez-vous à. Configuration de l'accès VPC pour que les applications EMR sans serveur se connectent aux données

- Si votre cluster Amazon Redshift ou votre groupe de travail Amazon Redshift Serverless provisionné est accessible au public, spécifiez un ou plusieurs sous-réseaux privés auxquels une passerelle NAT est attachée lorsque vous créez des applications EMR Serverless.
- Si votre cluster Amazon Redshift ou votre groupe de travail Amazon Redshift Serverless provisionné n'est pas accessible au public, vous devez créer un point de terminaison VPC géré par Amazon Redshift pour votre cluster Amazon Redshift, comme décrit dans. Configuration de l'accès VPC pour que les applications EMR sans serveur se connectent aux données Vous pouvez également créer votre groupe de travail Amazon Redshift Serverless comme décrit dans la section Connexion à Amazon Redshift Serverless du guide de gestion Amazon Redshift. Vous devez associer votre cluster ou votre sous-groupe aux sous-réseaux privés que vous spécifiez lorsque vous créez votre application EMR Serverless.

> **Note**
>
> Si vous utilisez l'authentification basée sur IAM et qu'aucune passerelle NAT n'est attachée à vos sous-réseaux privés pour l'application EMR Serverless, vous devez également créer un point de terminaison VPC sur ces sous-réseaux pour Amazon Redshift ou Amazon Redshift Serverless. De cette façon, le pilote JDBC peut récupérer les informations d'identification.

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 avec 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::account-id:role/role-name"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name") \
    .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", "table-name-copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .mode("error") \
    .save()
```

SparkSQL

PySpark À utiliser pour lire et écrire des exemples de données depuis et vers une base de données Amazon Redshift avec 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::account-id:role/role-name"
```

```
bucket = "s3://path/for/temp/data"
tableName = "table-name" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {table-name} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{table-name}', 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(table-name, overwrite=False)
df = spark.sql(f"SELECT * FROM {table-name}")
df.show()
```

Considérations et limites relatives à l'utilisation du connecteur Spark

- Nous vous conseillons d'activer le protocole SSL pour la connexion JDBC entre Spark on Amazon EMR et Amazon Redshift.
- Nous vous suggérons de gérer les informations d'identification du cluster Amazon Redshift dans le AWS Secrets Manager cadre des meilleures pratiques. Reportez-vous à AWS Secrets Manager la section Utiliser pour récupérer les informations d'identification permettant de vous connecter à Amazon Redshift pour un exemple.
- Nous vous suggérons de transmettre un rôle IAM avec le paramètre `aws_iam_role` 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 suggérons de bloquer l'accès public au cluster Amazon Redshift.
  - Nous vous conseillons d'activer la journalisation des audits Amazon Redshift.
  - Nous vous conseillons d'activer le chiffrement au repos d'Amazon Redshift.
- Tenez compte des recommandations suivantes pour Amazon S3 :

- Nous vous conseillons de bloquer l'accès public aux compartiments Amazon S3.
- Nous vous conseillons d'utiliser le chiffrement côté serveur Amazon S3 pour chiffrer les compartiments Amazon S3 utilisés.
- Nous vous suggérons d'utiliser les politiques de cycle de vie d'Amazon S3 pour définir les règles de conservation pour le 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 les méthodes d'authentification suivantes de Spark vers Amazon S3 :
  - Configuration des clés AWS d'accès dans la classification `hadoop-env` de configuration
  - Encodage des clés AWS d'accès dans l'`tempdir`URI

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

Connexion à DynamoDB avec Amazon EMR Serverless

Dans ce didacticiel, vous chargez un sous-ensemble de données du Bureau des noms géographiques des États-Unis d'Amérique dans un compartiment Amazon S3, puis vous utilisez Hive ou Spark sur Amazon EMR Serverless pour copier les données dans une table Amazon DynamoDB à des fins d'interrogation.

Étape 1 : télécharger des données dans un compartiment Amazon S3

Pour créer un compartiment Amazon S3, suivez les instructions de la section Création d'un compartiment dans le guide de l'utilisateur de la console Amazon Simple Storage Service. Remplacez les références *`amzn-s3-demo-bucket`* à par le nom du bucket que vous venez de créer. Votre application EMR Serverless est maintenant prête à exécuter des tâches.

1. Téléchargez l'exemple d'archive de données `features.zip` à l'aide de la commande suivante.

```
wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
features.zip
```

2. Extrayez le `features.txt` fichier de l'archive et accédez aux premières lignes du fichier :

```
unzip features.zip
head features.txt
```

Le résultat devrait ressembler à ce qui suit.

```
1535908|Big Run|Stream|WV|38.6370428|-80.8595469|794
875609|Constable Hook|Cape|NJ|40.657881|-74.0990309|7
1217998|Gooseberry Island|Island|RI|41.4534361|-71.3253284|10
26603|Boone Moore Spring|Spring|AZ|34.0895692|-111.410065|3681
1506738|Missouri Flat|Flat|WA|46.7634987|-117.0346113|2605
1181348|Minnow Run|Stream|PA|40.0820178|-79.3800349|1558
1288759|Hunting Creek|Stream|TN|36.343969|-83.8029682|1024
533060|Big Charles Bayou|Bay|LA|29.6046517|-91.9828654|0
829689|Greenwood Creek|Stream|NE|41.596086|-103.0499296|3671
541692|Button Willow Island|Island|LA|31.9579389|-93.0648847|98
```

Les champs de chaque ligne indiquent un identifiant unique, un nom, un type d'élément naturel, un état, une latitude en degrés, une longitude en degrés et une hauteur en pieds.

3. Chargez vos données sur Amazon S3

```
aws s3 cp features.txt s3://amzn-s3-demo-bucket/features/
```

Étape 2 : Création d'une table Hive

Utilisez Apache Spark ou Hive pour créer une nouvelle table Hive contenant les données téléchargées dans Amazon S3.

Spark

Pour créer une table Hive avec Spark, exécutez la commande suivante.

```
import org.apache.spark.sql.SparkSession

val sparkSession = SparkSession.builder().enableHiveSupport().getOrCreate()

sparkSession.sql("CREATE TABLE hive_features \
    (feature_id BIGINT, \
```

```
    feature_name STRING, \
    feature_class STRING, \
    state_alpha STRING, \
    prim_lat_dec DOUBLE, \
    prim_long_dec DOUBLE, \
    elev_in_ft BIGINT) \
    ROW FORMAT DELIMITED \
    FIELDS TERMINATED BY '|' \
    LINES TERMINATED BY '\n' \
    LOCATION 's3://amzn-s3-demo-bucket/features';")
```

Vous disposez désormais d'une table Hive remplie avec les données du `features.txt` fichier. Pour vérifier que vos données figurent dans la table, exécutez une requête SQL Spark comme indiqué dans l'exemple suivant.

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Pour créer une table Hive avec Hive, exécutez la commande suivante.

```
CREATE TABLE hive_features
    (feature_id             BIGINT,
    feature_name            STRING ,
    feature_class           STRING ,
    state_alpha             STRING,
    prim_lat_dec            DOUBLE ,
    prim_long_dec           DOUBLE ,
    elev_in_ft              BIGINT)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '|'
    LINES TERMINATED BY '\n'
    LOCATION 's3://amzn-s3-demo-bucket/features';
```

Vous disposez désormais d'une table Hive qui contient les données du `features.txt` fichier. Pour vérifier que vos données figurent dans la table, exécutez une requête HiveQL, comme indiqué dans l'exemple suivant.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

Étape 3 : Copier les données dans DynamoDB

Utilisez Spark ou Hive pour copier des données dans une nouvelle table DynamoDB.

Spark

Pour copier les données de la table Hive que vous avez créée à l'étape précédente dans DynamoDB, suivez les étapes 1 à 3 de la section Copier les données dans DynamoDB. Cela crée une nouvelle table DynamoDB appelée. `Features` Vous pouvez ensuite lire les données directement depuis le fichier texte et les copier dans votre table DynamoDB, comme le montre l'exemple suivant.

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue
import org.apache.hadoop.dynamodb.DynamoDBItemWritable
import org.apache.hadoop.dynamodb.read.DynamoDBInputFormat
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred.JobConf
import org.apache.spark.SparkContext

import scala.collection.JavaConverters._

object EmrServerlessDynamoDbTest {

    def main(args: Array[String]): Unit = {

        jobConf.set("dynamodb.input.tableName", "Features")
        jobConf.set("dynamodb.output.tableName", "Features")
        jobConf.set("dynamodb.region", "region")

        jobConf.set("mapred.output.format.class",
 "org.apache.hadoop.dynamodb.write.DynamoDBOutputFormat")
        jobConf.set("mapred.input.format.class",
 "org.apache.hadoop.dynamodb.read.DynamoDBInputFormat")

        val rdd = sc.textFile("s3://amzn-s3-demo-bucket/ddb-connector/")
            .map(row => {
                val line = row.split("\\|")
                val item = new DynamoDBItemWritable()

                val elevInFt = if (line.length > 6) {
                    new AttributeValue().withN(line(6))
                } else {
                    new AttributeValue().withNULL(true)
```

```
                }

                item.setItem(Map(
                    "feature_id" -> new AttributeValue().withN(line(0)),
                    "feature_name" -> new AttributeValue(line(1)),
                    "feature_class" -> new AttributeValue(line(2)),
                    "state_alpha" -> new AttributeValue(line(3)),
                    "prim_lat_dec" -> new AttributeValue().withN(line(4)),
                    "prim_long_dec" -> new AttributeValue().withN(line(5)),
                    "elev_in_ft" -> elevInFt)
                    .asJava)
                (new Text(""), item)
        })
        rdd.saveAsHadoopDataset(jobConf)
    }
}
```

Hive

Pour copier les données de la table Hive que vous avez créée à l'étape précédente dans DynamoDB, suivez les instructions de la section Copier les données dans DynamoDB.

Étape 4 : demander des données à partir de DynamoDB

Utilisez Spark ou Hive pour interroger votre table DynamoDB.

Spark

Pour interroger les données de la table DynamoDB que vous avez créée à l'étape précédente, utilisez Spark SQL ou l'API Spark. MapReduce

Example— Interrogez votre table DynamoDB avec Spark SQL

La requête SQL Spark suivante renvoie une liste de tous les types de fonctionnalités par ordre alphabétique.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

La requête SQL Spark suivante renvoie une liste de tous les lacs commençant par la lettre M.

```
val dataFrame = sparkSession.sql("SELECT feature_name, state_alpha \
    FROM ddb_features \
    WHERE feature_class = 'Lake' \
    AND feature_name LIKE 'M%' \
    ORDER BY feature_name;")
```

La requête SQL Spark suivante renvoie une liste de tous les états comportant au moins trois entités situées à plus d'un kilomètre.

```
val dataFrame = sparkSession.dql("SELECT state_alpha, feature_class, COUNT(*) \
    FROM ddb_features \
    WHERE elev_in_ft > 5280 \
    GROUP by state_alpha, feature_class \
    HAVING COUNT(*) >= 3 \
    ORDER BY state_alpha, feature_class;")
```

Example— Interrogez votre table DynamoDB avec l'API Spark MapReduce

La MapReduce requête suivante renvoie une liste de tous les types de fonctionnalités par ordre alphabétique.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .map(pair => pair._2.get("feature_class").getS)
    .distinct()
    .sortBy(value => value)
    .toDF("feature_class")
```

La MapReduce requête suivante renvoie une liste de tous les lacs commençant par la lettre M.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .filter(pair => "Lake".equals(pair._2.get("feature_class").getS))
    .filter(pair => pair._2.get("feature_name").getS.startsWith("M"))
    .map(pair => (pair._2.get("feature_name").getS,
 pair._2.get("state_alpha").getS))
    .sortBy(_._1)
    .toDF("feature_name", "state_alpha")
```

La MapReduce requête suivante renvoie une liste de tous les états comportant au moins trois entités situées à plus d'un kilomètre.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => pair._2.getItem)
    .filter(pair => pair.get("elev_in_ft").getN != null)
    .filter(pair => Integer.parseInt(pair.get("elev_in_ft").getN) > 5280)
    .groupBy(pair => (pair.get("state_alpha").getS, pair.get("feature_class").getS))
    .filter(pair => pair._2.size >= 3)
    .map(pair => (pair._1._1, pair._1._2, pair._2.size))
    .sortBy(pair => (pair._1, pair._2))
    .toDF("state_alpha", "feature_class", "count")
```

Hive

Pour interroger les données de la table DynamoDB que vous avez créée à l'étape précédente, suivez les instructions de la section Interroger les données de la table DynamoDB.

Configurer l'accès intercompte

Pour configurer l'accès entre comptes pour EMR Serverless, procédez comme suit. Dans l'exemple, `AccountA` il s'agit du compte sur lequel vous avez créé votre application Amazon EMR Serverless et `AccountB` du compte sur lequel se trouve votre Amazon DynamoDB.

1. Créez une table DynamoDB dans. `AccountB` Pour plus d'informations, reportez-vous à Étape 1 : Création d'une table.
2. Créez un rôle `Cross-Account-Role-B` IAM permettant d'accéder à `AccountB` la table DynamoDB.

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

   b. Choisissez Rôles, puis créez un nouveau rôle appelé`Cross-Account-Role-B`. Pour plus d'informations sur la création de rôles IAM, reportez-vous à la section Création de rôles IAM dans le guide de l'utilisateur.

   c. Créez une politique IAM qui accorde les autorisations d'accès à la table DynamoDB entre comptes. Attachez ensuite la politique IAM au `Cross-Account-Role-B`.

      Voici une politique qui autorise l'accès à une table DynamoDB`CrossAccountTable`.

d. Modifiez la relation de confiance du rôle `Cross-Account-Role-B`.

Pour configurer la relation de confiance pour le rôle, choisissez l'onglet Relations de confiance dans la console IAM pour le rôle que vous avez créé à l'étape 2 : Cross-Account-Role-B.

Sélectionnez Modifier la relation de confiance, puis ajoutez le document de politique suivant. Ce document permet `AccountA` à `Job-Execution-Role-A` in d'assumer ce `Cross-Account-Role-B` rôle.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/Job-Execution-Role-A",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

e. Accordez `Job-Execution-Role-A` `AccountA` avec `- STS Assume role` les autorisations d'assumer`Cross-Account-Role-B`.

Dans la console IAM pour Compte AWS `AccountA`, sélectionnez`Job-Execution-Role-A`. 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::123456789012:role/Cross-Account-Role-B"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

f. Définissez la `dynamodb.customAWSCredentialsProvider` propriété avec une valeur comme `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` dans la classification des sites principaux. Définissez la variable d'environnement `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` avec la valeur ARN de`Cross-Account-Role-B`.

3. Exécutez le job Spark ou Hive en utilisant`Job-Execution-Role-A`.

Considérations

Notez ces comportements et limitations lorsque vous utilisez le connecteur DynamoDB avec Apache Spark ou Apache Hive.

Considérations relatives à l'utilisation du connecteur DynamoDB avec Apache Spark

- Spark SQL ne prend pas en charge la création d'une table Hive avec l'option storage-handler. Pour plus d'informations, reportez-vous à la section Spécification du format de stockage pour les tables Hive dans la documentation d'Apache Spark.
- Spark SQL ne prend pas en charge l'`STORED BY`opération avec le gestionnaire de stockage. Si vous souhaitez interagir avec une table DynamoDB via une table Hive externe, utilisez d'abord Hive pour créer la table.
- Pour convertir une requête en requête DynamoDB, le connecteur DynamoDB utilise le prédicat pushdown. Le prédicat pushdown filtre les données en fonction d'une colonne mappée à la clé de partition d'une table DynamoDB. Le pushdown des prédicats ne fonctionne que lorsque vous utilisez le connecteur avec Spark SQL, et non avec l' MapReduce API.

Considérations relatives à l'utilisation du connecteur DynamoDB avec Apache Hive

Réglage du nombre maximum de mappeurs

- Si vous utilisez la `SELECT` requête pour lire les données d'une table Hive externe mappée à DynamoDB, le nombre de tâches cartographiques sur EMR Serverless est calculé comme le débit de lecture total configuré pour la table DynamoDB, divisé par le débit par tâche cartographique. Le débit par défaut par tâche cartographique est de 100.
- La tâche Hive peut utiliser le nombre de tâches cartographiques au-delà du nombre maximum de conteneurs configurés par application EMR Serverless, en fonction du débit de lecture configuré pour DynamoDB. En outre, une requête Hive de longue durée peut consommer toute la capacité de lecture allouée à la table DynamoDB. Cela a un impact négatif sur les autres utilisateurs.
- Vous pouvez utiliser cette `dynamodb.max.map.tasks` propriété pour définir une limite supérieure pour les tâches cartographiques. Vous pouvez également utiliser cette propriété pour ajuster la quantité de données lues par chaque tâche cartographique en fonction de la taille du conteneur de tâches.
- Vous pouvez définir la `dynamodb.max.map.tasks` propriété au niveau de la requête Hive ou dans la `hive-site` classification de la start-job-run commande. Cette valeur doit être supérieure ou égale à 1. Lorsque Hive traite votre requête, la tâche Hive qui en résulte n'utilise que les valeurs indiquées `dynamodb.max.map.tasks` lors de la lecture dans la table DynamoDB.

Réglage du débit d'écriture par tâche

- Le débit d'écriture par tâche sur EMR Serverless est calculé comme le débit d'écriture total configuré pour une table DynamoDB, divisé par la valeur de la propriété. `mapreduce.job.maps` Pour Hive, la valeur par défaut de cette propriété est 2. Ainsi, les deux premières tâches de la phase finale d'une tâche Hive peuvent consommer tout le débit d'écriture. Cela entraîne une limitation des écritures d'autres tâches dans le même travail ou dans d'autres tâches.
- Pour éviter la limitation de l'écriture, définissez la valeur de la `mapreduce.job.maps` propriété en fonction du nombre de tâches de l'étape finale ou du débit d'écriture que vous souhaitez allouer par tâche. Définissez cette propriété dans la `mapred-site` classification de la start-job-run commande sur EMR Serverless.

Sécurité

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous utilisez en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon EMR Serverless, reportez-vous à la section AWS Services concernés par le programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre entreprise, ainsi que la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon EMR Serverless. Les rubriques expliquent comment configurer Amazon EMR Serverless et comment utiliser d'autres AWS services pour atteindre vos objectifs de sécurité et de conformité.

Rubriques

- Bonnes pratiques de sécurité pour Amazon EMR Serverless
- Protection des données
- Identity and Access Management (IAM) dans Amazon EMR Serverless
- Propagation d'identité fiable
- Utilisation de Lake Formation avec EMR sans serveur
- Chiffrement inter-travailleurs
- Chiffrement de disque avec KMS CMK
- Secrets Manager pour la protection des données avec EMR Serverless
- Utilisation des autorisations d'accès Amazon S3 avec EMR sans serveur

- Journalisation des appels d'API sans serveur Amazon EMR à l'aide de AWS CloudTrail
- Validation de conformité pour Amazon EMR Serverless
- Résilience dans Amazon EMR Serverless
- Sécurité de l'infrastructure dans Amazon EMR Serverless
- Analyse de configuration et de vulnérabilité dans Amazon EMR Serverless

Bonnes pratiques de sécurité pour Amazon EMR Serverless

Amazon EMR Serverless fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques 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.

Application du principe du moindre privilège

EMR Serverless fournit une politique d'accès granulaire pour les applications utilisant des rôles IAM, tels que les rôles d'exécution. Nous suggérons 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.

Isolation du code d'application non fiable

EMR Serverless crée une isolation complète du réseau entre les tâches appartenant aux différentes applications EMR Serverless. Dans les cas où l'isolation au niveau des tâches est souhaitée, envisagez d'isoler les tâches dans différentes applications EMR sans serveur.

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 les applications EMR sans serveur.

Protection des données

Le modèle de responsabilité AWS partagée s'applique à la protection des données dans Amazon EMR Serverless. 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 relève de votre responsabilité. Ce contenu inclut la configuration de la sécurité et les tâches de gestion pour les AWS services que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la 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 suggérons 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 suggérons le protocole 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 sans serveur Amazon EMR 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 disponibles, reportez-vous à la norme fédérale de traitement de l'information (FIPS) 140-2.

Nous vous conseillons vivement de ne jamais saisir d'informations d'identification sensibles, telles que les numéros de compte de vos clients, dans des champs libres tels que le champ Nom. Cela inclut lorsque vous travaillez avec Amazon EMR Serverless ou d'autres AWS services à l'aide de la

console, de l'API ou. AWS CLI AWS SDKs Toutes les données que vous entrez dans Amazon EMR Serverless 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, reportez-vous à la section AWS KMS Tarification.

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

Chaque application EMR Serverless utilise une version de version spécifique, qui inclut EMRFS (EMR File System). Le chiffrement Amazon S3 fonctionne avec les objets du système de fichiers EMR (EMRFS) lus et écrits sur Amazon S3. Vous pouvez spécifier le chiffrement côté serveur (SSE) ou le chiffrement côté client (CSE) Amazon S3 comme mode de chiffrement par défaut lorsque vous activez le chiffrement au repos. Vous pouvez éventuellement 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. Si vous utilisez Amazon S3 CSE avec des clés gérées par le client, votre rôle d'exécution utilisé pour exécuter des tâches dans une application EMR sans serveur doit avoir accès à la clé. Pour obtenir des informations détaillées sur le chiffrement Amazon S3, reportez-vous à la section Protection des données à l'aide du chiffrement du manuel Amazon Simple Storage Service Developer Guide.

> **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, reportez-vous à la section AWS KMS Tarification.

Chiffrement côté serveur sur Amazon S3

Le chiffrement est configuré par défaut pour tous les compartiments Amazon S3, et tous les nouveaux objets chargés dans un compartiment S3 sont automatiquement chiffrés au repos. Amazon S3 chiffre les données au niveau de l'objet lorsqu'il écrit les données sur le disque et les déchiffre lors de l'accès. Pour plus d'informations sur SSE, reportez-vous à la section Protection des données à l'aide du chiffrement côté serveur dans le manuel Amazon Simple Storage Service Developer Guide.

Vous pouvez choisir entre deux systèmes de gestion de clés différents lorsque vous spécifiez SSE dans Amazon EMR Serverless :

- SSE-S3 : Amazon S3 gère les clés pour vous. Aucune configuration supplémentaire n'est requise sur EMR Serverless.
- SSE-KMS - Vous utilisez un AWS KMS key pour configurer des politiques adaptées à EMR Serverless. Aucune configuration supplémentaire n'est requise sur EMR Serverless.

Pour utiliser AWS KMS le chiffrement des données que vous écrivez sur Amazon S3, deux options s'offrent à vous lorsque vous utilisez l'`StartJobRun`API. Vous pouvez soit activer le chiffrement pour tout ce que vous écrivez sur Amazon S3, soit activer le chiffrement pour les données que vous écrivez dans un compartiment spécifique. Pour plus d'informations sur l'`StartJobRun`API, reportez-vous au manuel EMR Serverless API Reference.

Pour activer le AWS KMS chiffrement de toutes les données que vous écrivez sur Amazon S3, utilisez les commandes suivantes lorsque vous appelez l'`StartJobRun`API.

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Pour activer le AWS KMS chiffrement des données que vous écrivez dans un compartiment spécifique, utilisez les commandes suivantes lorsque vous appelez l'`StartJobRun`API.

```
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-bucket1>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-
bucket1>.serverSideEncryption.kms.keyId=<kms-id>
```

Le SSE avec clés fournies par le client (SSE-C) n'est pas disponible pour une utilisation avec EMR Serverless.

Chiffrement côté client sur Amazon S3

Avec le chiffrement côté client Amazon S3, le chiffrement et le déchiffrement Amazon S3 ont lieu dans le client EMRFS disponible sur chaque version d'Amazon EMR. 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. Si vous utilisez Amazon S3 CSE avec des clés gérées par le client, votre rôle d'exécution utilisé pour exécuter des tâches dans une application EMR sans serveur doit avoir accès à la clé. Des frais KMS supplémentaires peuvent s'appliquer. Pour plus d'informations sur ces différences, reportez-vous à la section Protection des données à l'aide du chiffrement côté client du manuel Amazon Simple Storage Service Developer Guide.

Chiffrement de disque local

Les données stockées dans un stockage éphémère sont cryptées à l'aide de clés appartenant au service à l'aide de l'algorithme cryptographique AES-256 standard du secteur.

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, reportez-vous à Rotation des clés gérées par le client.

Chiffrement en transit

Les fonctionnalités de chiffrement spécifiques aux applications suivantes sont disponibles avec Amazon EMR Serverless :

- Spark
  - Par défaut, la communication entre les pilotes Spark et les exécuteurs est authentifiée et interne. La communication RPC entre les pilotes et les exécuteurs est cryptée.
- Hive
  - La communication entre le métastore AWS Glue et les applications EMR Serverless s'effectue via le protocole TLS.

Vous devez autoriser uniquement les connexions chiffrées via HTTPS (TLS) conformément à la SecureTransport condition aws : des politiques IAM du compartiment Amazon S3.

Identity and Access Management (IAM) dans Amazon EMR Serverless

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources Amazon EMR Serverless. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

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 des problèmes d'identité et d'accès sans serveur Amazon EMR)
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir Identity and Access Management (IAM) dans Amazon EMR Serverless)
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir Exemples de politiques basées sur l'identité pour EMR Serverless)

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 d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

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.

Comment EMR Serverless fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon EMR Serverless, découvrez quelles fonctionnalités IAM peuvent être utilisées avec Amazon EMR Serverless.

Utilisation des fonctionnalités IAM avec EMR Serverless

Fonctionnalité IAM	Support sans serveur Amazon EMR
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition d'une politique	Non
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 EMR Serverless et les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, reportez-vous aux AWS services compatibles avec IAM dans le guide de l'utilisateur IAM.

Politiques basées sur l'identité pour EMR Serverless

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 EMR Serverless

Pour accéder à des exemples de politiques basées sur l'identité Amazon EMR Serverless, reportez-vous à. Exemples de politiques basées sur l'identité pour EMR Serverless

Politiques basées sur les ressources dans EMR Serverless

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 stratégiques pour EMR Serverless

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 consulter la liste des actions EMR sans serveur, reportez-vous à la section Actions, ressources et clés de condition pour Amazon EMR Serverless dans le Service Authorization Reference.

Les actions de stratégie dans EMR Serverless utilisent le préfixe suivant avant l'action.

```
emr-serverless
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Pour accéder à des exemples de politiques basées sur l'identité Amazon EMR Serverless, reportez-vous à. Exemples de politiques basées sur l'identité pour EMR Serverless

Ressources relatives aux politiques relatives à l'EMR sans serveur

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 Serverless et leurs ARNs, consultez la section Ressources définies par Amazon EMR Serverless dans le Service Authorization Reference. Pour savoir quelles actions spécifient l'ARN de chaque ressource, reportez-vous à la section Actions, ressources et clés de condition pour Amazon EMR Serverless.

Pour accéder à des exemples de politiques basées sur l'identité Amazon EMR Serverless, reportez-vous à. Exemples de politiques basées sur l'identité pour EMR Serverless

Clés de conditions de politique pour EMR Serverless

Support des clés d'état des politiques

Prend en charge les clés de condition de politique spécifiques au service	Non

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 Amazon EMR Serverless et pour savoir quelles actions et ressources vous pouvez utiliser une clé de condition, reportez-vous à la section Actions, ressources et clés de condition pour Amazon EMR Serverless dans le Service Authorization Reference.

Toutes les actions Amazon EC2 prennent en charge les clés de condition `aws:RequestedRegion` et `ec2:Region`. Pour plus d'informations, reportez-vous à Exemple : restriction de l'accès à une région spécifique.

Listes de contrôle d'accès (ACLs) dans EMR Serverless

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 EMR Serverless

Support du contrôle d'accès basé sur les attributs (ABAC)

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 EMR Serverless

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 principales interservices pour EMR Serverless

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.

Rôles de service pour EMR Serverless

Prend en charge les fonctions de service	Non

Rôles liés à un service pour EMR Serverless

Prend en charge les rôles liés à un service.	Oui

Pour plus de détails sur la création ou la gestion des rôles liés à un service, reportez-vous aux AWS services qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Cliquez sur le lien Oui pour accéder à la documentation des rôles liés au service pour ce service.

Utilisation de rôles liés à un service pour EMR Serverless

Amazon EMR Serverless utilise des rôles liés à un service Gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à EMR Serverless. Les rôles liés à un service sont prédéfinis par EMR Serverless et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'EMR Serverless, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. EMR Serverless définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul EMR Serverless 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. Cela protège vos ressources EMR Serverless, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés aux services, reportez-vous à la section AWS Services qui fonctionnent avec IAM et recherchez les services dont la

valeur est Oui dans la colonne Rôles liés aux services. Cliquez sur Oui avec un lien pour accéder à la documentation des rôles liés au service correspondant à ce service.

Autorisations de rôle liées à un service pour EMR Serverless

EMR Serverless utilise le rôle lié au service nommé AWSServiceRoleForAmazonEMRServerlesspour lui permettre d'appeler en votre nom. AWS APIs

Le rôle AWSService RoleForAmazon EMRServerless lié à un service fait confiance aux services suivants pour assumer le rôle :

- `ops.emr-serverless.amazonaws.com`

La politique d'autorisations de rôle nommée `AmazonEMRServerlessServiceRolePolicy` permet à EMR Serverless d'effectuer les actions suivantes sur les ressources spécifiées.

> **Note**
>
> Le contenu des politiques gérées étant modifié, il est possible que la politique présentée ici ne soit plus à jour. Consultez la up-to-date politique la plus importante d'Amazon EMRServerless ServiceRolePolicy dans le AWS Management Console.

- Action : `ec2:CreateNetworkInterface`
- Action : `ec2:DeleteNetworkInterface`
- Action : `ec2:DescribeNetworkInterfaces`
- Action : `ec2:DescribeSecurityGroups`
- Action : `ec2:DescribeSubnets`
- Action : `ec2:DescribeVpcs`
- Action : `ec2:DescribeDhcpOptions`
- Action : `ec2:DescribeRouteTables`
- Action : `cloudwatch:PutMetricData`

Voici la `AmazonEMRServerlessServiceRolePolicy` politique complète.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EC2PolicyStatement",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeRouteTables"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CloudWatchPolicyStatement",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/EMRServerless",
            "AWS/Usage"
          ]
        }
      }
    }
  ]
}
```

La politique de confiance suivante est attachée à ce rôle pour permettre au principal EMR Serverless d'assumer ce rôle.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/aws-service-role/emr-
serverless.amazonaws.com/AWSServiceRoleForEMRServerless",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, reportez-vous à la section Autorisations relatives aux rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour EMR Serverless

Vous n’avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez une nouvelle application EMR Serverless dans (à l' AWS Management Console aide d'EMR Studio), ou dans l' AWS CLI API AWS , EMR Serverless crée le rôle lié au service pour vous. 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 créer le rôle AWSService RoleForAmazon EMRServerless lié à un service à l'aide d'IAM

Ajoutez l'instruction suivante à la politique d'autorisations pour l'entité IAM qui doit créer le rôle lié à un service.

```
{
    "Effect": "Allow",
```

```
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Si vous supprimez ce rôle lié à un service, puis que vous devez le créer à nouveau, suivez le même processus pour recréer le rôle dans votre compte. Lorsque vous créez une nouvelle application EMR Serverless, EMR Serverless crée à nouveau le rôle lié au service pour vous.

Vous pouvez également utiliser la console IAM pour créer un rôle lié à un service avec le cas d'utilisation EMR Serverless. Dans l'API AWS CLI ou dans l' AWS API, créez un rôle lié à un service avec le nom du `ops.emr-serverless.amazonaws.com` service. Pour plus d'informations, reportez-vous à la section Création d'un rôle lié à un service dans le Guide de l'utilisateur IAM. Si vous supprimez ce rôle lié à un service, utilisez le même processus pour le créer à nouveau.

Modification d'un rôle lié à un service pour EMR Serverless

EMR Serverless ne vous permet pas de modifier le rôle AWSService RoleForAmazon EMRServerless lié au service car diverses entités peuvent y faire référence. Vous ne pouvez pas modifier la politique IAM AWS détenue par le rôle lié au service EMR Serverless, car elle contient toutes les autorisations nécessaires dont EMR Serverless a besoin. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM.

Pour modifier la description du rôle AWSService RoleForAmazon EMRServerless lié à un service à l'aide d'IAM

Ajoutez l'instruction suivante à la stratégie d'autorisation de l'entité IAM qui doit modifier la description d'un rôle lié à un service.

```
{
    "Effect": "Allow",
    "Action": [
        "iam: UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
```

```
}
```

Pour plus d'informations, reportez-vous à la section Modification d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Suppression d'un rôle lié à un service pour EMR Serverless

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service nécessitant un rôle lié à un service, nous vous suggérons de supprimer ce rôle. Ainsi, vous n'avez pas d'entité inutilisée qui n'est pas activement surveillée ou maintenue. Toutefois, supprimez toutes les applications EMR sans serveur dans toutes les régions avant de supprimer le rôle lié à un service.

> **Note**
>
> Si le service EMR Serverless utilise le rôle lorsque vous essayez de supprimer les ressources associées au rôle, la suppression risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour supprimer le rôle AWSService RoleForAmazon EMRServerless lié à un service à l'aide d'IAM

Ajoutez la déclaration suivante à la politique d'autorisations pour l'entité IAM qui doit supprimer un rôle lié à un service.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

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 AWSService RoleForAmazon EMRServerless service. Pour plus d'informations, reportez-vous à la section 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 EMR sans serveur

EMR Serverless prend en charge l'utilisation de rôles liés au service dans toutes les régions où le service est disponible. Pour plus d'informations, reportez-vous à la section AWS Régions et points de terminaison.

Rôles d'exécution des tâches pour Amazon EMR Serverless

Vous pouvez spécifier les autorisations de rôle IAM que l'exécution d'une tâche EMR sans serveur peut assumer lorsque vous appelez d'autres services en votre nom. Cela inclut l'accès à Amazon S3 pour toutes les sources de données, les cibles, ainsi que pour d'autres AWS ressources telles que les clusters Amazon Redshift et les tables DynamoDB. Pour en savoir plus sur la création d'un rôle, reportez-vous àCréation d'un rôle d'exécution de tâches.

Exemples de politiques d'exécution

Vous pouvez associer une politique d'exécution, telle que la suivante, à un rôle d'exécution de tâche. La politique d'exécution des tâches suivante permet :

- Accès en lecture aux compartiments Amazon S3 contenant des exemples d'EMR.
- Accès complet aux compartiments S3.
- Créez et lisez l'accès au catalogue de données AWS Glue.

Pour ajouter l'accès à d'autres AWS ressources telles que DynamoDB, vous devez inclure les autorisations correspondantes dans la politique lors de la création du rôle d'exécution.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.elasticmapreduce",
```

```
        "arn:aws:s3:::*.elasticmapreduce/*"
      ]
    },
    {
      "Sid": "FullAccessToS3Bucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Transmettez les privilèges liés aux rôles

Vous pouvez associer des politiques d'autorisation IAM au rôle d'un utilisateur pour lui permettre de transmettre uniquement les rôles approuvés. Cela permet aux administrateurs de contrôler quels utilisateurs peuvent transmettre des rôles d'exécution de tâches spécifiques aux tâches EMR Serverless. Pour en savoir plus sur la définition des autorisations, reportez-vous à la section Accorder à un utilisateur l'autorisation de transmettre un rôle à un AWS service.

Voici un exemple de politique qui permet de transmettre un rôle d'exécution de tâche au principal du service EMR Serverless.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

Politiques d'autorisation gérées associées aux rôles d'exécution

Lorsque vous soumettez des exécutions de tâches à EMR sans serveur via la console EMR Studio, vous choisissez une étape au cours de laquelle vous choisissez un rôle Runtime à associer à votre application. Il est important de connaître les politiques gérées sous-jacentes associées à chaque sélection dans la console. Les trois sélections sont les suivantes :

1. Tous les compartiments : lorsque vous choisissez cette option, elle spécifie la politique FullAccess AWS gérée par AmazonS3, qui fournit un accès complet à tous les compartiments.
2. Buckets spécifiques : cela spécifie l'identifiant du nom de ressource Amazon (ARN) de chaque compartiment que vous choisissez. Aucune politique gérée sous-jacente n'est incluse.
3. Aucune — Aucune autorisation basée sur des politiques gérées n'est incluse.

Nous vous suggérons d'ajouter des compartiments spécifiques. Si vous choisissez tous les compartiments, n'oubliez pas qu'il définit un accès complet pour tous les compartiments.

Exemples de politiques d'accès utilisateur pour EMR Serverless

Vous pouvez définir des politiques précises pour vos utilisateurs en fonction des actions que vous souhaitez que chaque utilisateur effectue lorsqu'il interagit avec des applications EMR sans serveur. Les politiques suivantes sont des exemples susceptibles de vous aider à configurer les autorisations appropriées pour vos utilisateurs. Cette section se concentre uniquement sur les politiques EMR Serverless. Pour des exemples de politiques utilisateur d'EMR Studio, reportez-vous à la section Configurer les autorisations utilisateur d'EMR Studio. Pour plus d'informations sur la façon d'associer des politiques aux utilisateurs IAM (principaux), reportez-vous à la section Gestion des politiques IAM du Guide de l'utilisateur IAM.

Politique relative aux utilisateurs expérimentés

Pour accorder toutes les actions requises pour EMR Serverless, créez et associez une `AmazonEMRServerlessFullAccess` politique à l'utilisateur, au rôle ou au groupe IAM requis.

Voici un exemple de politique qui permet aux utilisateurs expérimentés de créer et de modifier des applications EMR sans serveur, ainsi que d'effectuer d'autres actions telles que la soumission et le débogage de tâches. Il révèle toutes les actions requises par EMR Serverless pour les autres services.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication",
        "emr-serverless:UpdateApplication",
        "emr-serverless:DeleteApplication",
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StopApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
```

```
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Lorsque vous activez la connectivité réseau à votre VPC, les applications EMR Serverless créent des interfaces réseau élastiques (ENI) Amazon EC2 pour communiquer avec les ressources du VPC. La politique suivante garantit que tout nouvel EC2 ENIs est créé uniquement dans le contexte des applications EMR sans serveur.

> **Note**
>
> Nous recommandons vivement de définir cette politique afin de garantir que les utilisateurs ne puissent pas créer d'EC2, ENIs sauf dans le cadre du lancement d'applications EMR sans serveur.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
        }
      }
```

```
    }
  ]
}
```

Si vous souhaitez restreindre l'accès EMR Serverless à certains sous-réseaux, vous pouvez étiqueter chaque sous-réseau avec une condition de balise. Cette politique IAM garantit que les applications EMR sans serveur ne peuvent créer de l' ENIs EC2 que dans les sous-réseaux autorisés.

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/KEY": "VALUE"
        }
    }
}
```

> **Important**
>
> Si vous êtes un administrateur ou un utilisateur expérimenté qui crée votre première application, vous devez configurer vos politiques d'autorisation pour vous permettre de créer un rôle lié à un service EMR Serverless. Pour en savoir plus, reportez-vous àUtilisation de rôles liés à un service pour EMR Serverless.

La politique IAM suivante vous permet de créer un rôle lié à un service EMR Serverless pour votre compte.

```
{
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
```

```
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

Politique relative aux ingénieurs de données

Voici un exemple de politique qui accorde aux utilisateurs des autorisations en lecture seule sur les applications EMR Serverless, ainsi que la possibilité de soumettre et de déboguer des tâches. Gardez à l'esprit que, si cette stratégie n'empêche pas explicitement certaines actions, une autre déclaration de stratégie peut toutefois être utilisée pour accorder l'accès aux actions spécifiées.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Utilisation de balises pour le contrôle d’accès

Vous pouvez utiliser les conditions des balises pour un contrôle d'accès précis. Par exemple, vous pouvez limiter le nombre d'utilisateurs à une seule équipe afin qu'ils ne puissent soumettre des tâches qu'aux applications EMR Serverless étiquetées avec le nom de leur équipe.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "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 applications et aux exécutions de tâches en fonction des balises.

Les exemples suivants illustrent différents scénarios et manières d'utiliser les opérateurs de condition avec les clés de condition EMR Serverless. 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 interdisant les actions de balisage, reportez-vous àRefuser 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 avec les applications EMR sans serveur.

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 `dev` correspondre la valeur du département des balises. Si le département des balises n'a pas été ajouté à l'application ou ne contient pas la valeur`dev`, la politique ne s'applique pas et les actions ne sont pas autorisées par cette politique. Si aucune autre déclaration de politique n'autorise les actions, l'utilisateur ne peut travailler qu'avec les applications dotées de cette balise avec cette valeur.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSGetapplication"
    }
  ]
}
```

Vous pouvez également spécifier plusieurs valeurs de balise à l'aide d'un opérateur de condition. Par exemple, pour autoriser des actions sur des applications où la `department` balise contient la valeur `dev` ou `test` pour remplacer le bloc de condition dans l'exemple précédent par le suivant.

```
"Condition": {
        "StringEquals": {
          "emr-serverless: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 de l'application.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

La déclaration de politique suivante permet à un utilisateur de créer une application uniquement si celle-ci possède une `department` balise, qui peut contenir n'importe quelle valeur.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-east-1", "us-west-2"]
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

Refuser l'accès pour ajouter et supprimer des balises

Cette politique empêche un utilisateur d'ajouter ou de supprimer des balises dans les applications EMR Serverless dont la valeur n'est `department` pas la même. `dev`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": [
```

```
        "*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSTagresource"
    }
  ]
}
```

Exemples de politiques basées sur l'identité pour EMR Serverless

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources Amazon EMR Serverless. 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 Serverless, y compris le format du ARNs pour chacun des types de ressources, consultez la section Actions, ressources et clés de condition pour Amazon EMR Serverless dans la référence d'autorisation de service.

Rubriques

- Bonnes pratiques en matière de politiques
- Permettre aux utilisateurs d'accéder à leurs propres autorisations

Bonnes pratiques en matière de politiques

> **Note**
>
> EMR Serverless ne prend pas en charge les politiques gérées. La première pratique décrite ci-dessous ne s'applique donc pas.

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Amazon EMR Serverless 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.

Permettre aux utilisateurs d'accéder à 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": "*"
```

```
        }
    ]
}
```

Amazon EMR Serverless : mises à jour des politiques gérées AWS

Accédez aux informations relatives aux mises à jour des politiques AWS gérées pour Amazon EMR Serverless depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique des documents Amazon EMR Serverless.

Modifier	Description	Date
Amazon EMRServerless ServiceRolePolicy — Mise à jour d'une politique existante	Amazon EMR Serverless a ajouté le nouveau `SidCloudWatchPolicyStatement` et `EC2PolicyStatement` à la politique d'Amazon. EMRServerless ServiceRolePolicy	25 janvier 2024
Amazon EMRServerless ServiceRolePolicy — Mise à jour d'une politique existante	Amazon EMR Serverless a ajouté de nouvelles autorisations pour permettre à Amazon EMR Serverless de publier des statistiques de compte agrégées relatives à l'utilisation des vCPU dans l'espace de noms. `"AWS/Usage"`	20 avril 2023
Amazon EMR Serverless a commencé à suivre les modifications	Amazon EMR Serverless a commencé à suivre les modifications apportées à ses AWS politiques gérées.	20 avril 2023

Résolution des problèmes d'identité et d'accès sans serveur Amazon EMR

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon EMR Serverless et IAM.

Rubriques

Je ne suis pas autorisé à effectuer une action dans Amazon EMR Serverless

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, contactez 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 lorsque l'`mateojackson`utilisateur essaie d'utiliser la console pour accéder aux détails d'une *`my-example-widget`* ressource fictive mais ne dispose pas des `emr-serverless:`*`GetWidget`* autorisations fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless: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-serverless:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon EMR Serverless.

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 Serverless. 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 Serverless

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 Serverless prend en charge ces fonctionnalités, consultez. Identity and Access Management (IAM) dans Amazon EMR Serverless
- 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.

Je ne parviens pas à ouvrir le serveur Live UI/Spark History depuis EMR Studio pour déboguer ma tâche ou une erreur d'API se produit lorsque j'essaie d'obtenir des journaux en utilisant **`get-dashboard-for-job-run`**

Si vous utilisez le stockage géré EMR sans serveur pour la journalisation et que votre application EMR sans serveur se trouve dans un sous-réseau privé avec des points de terminaison VPC pour Amazon S3 et que vous attachez une politique de point de terminaison pour contrôler l'accès, ajoutez les autorisations mentionnées dans la section Journalisation pour EMR sans serveur dans votre politique VPC au point de terminaison de passerelle S3 pour que EMR Serverless stocke et diffuse les journaux des applications.

Propagation d'identité fiable

Avec les versions 7.8.0 et supérieures d'Amazon EMR, vous pouvez propager les identités des utilisateurs depuis AWS IAM Identity Center vers des charges de travail interactives avec EMR Serverless via Apache Livy Endpoint. Les charges de travail interactives Apache Livy propageront davantage l'identité fournie aux services en aval tels qu'Amazon S3, Lake Formation et Amazon Redshift, permettant ainsi un accès sécurisé aux données via l'identité de l'utilisateur dans ces services en aval. Les sections suivantes fournissent une présentation conceptuelle, les prérequis et les étapes nécessaires pour lancer et propager l'identité aux charges de travail interactives avec EMR Serverless via Apache Livy Endpoint.

Présentation de

L'IAM Identity Center est l'approche recommandée pour l'authentification et l'autorisation du personnel AWS pour les organisations de toutes tailles et de tous types. Avec Identity Center, créez et gérez les identités des utilisateurs dans AWS ou connectez votre source d'identité existante, notamment Microsoft Active Directory, Okta, Ping Identity JumpCloud, Google Workspace et Microsoft Entra ID (anciennement Azure AD).

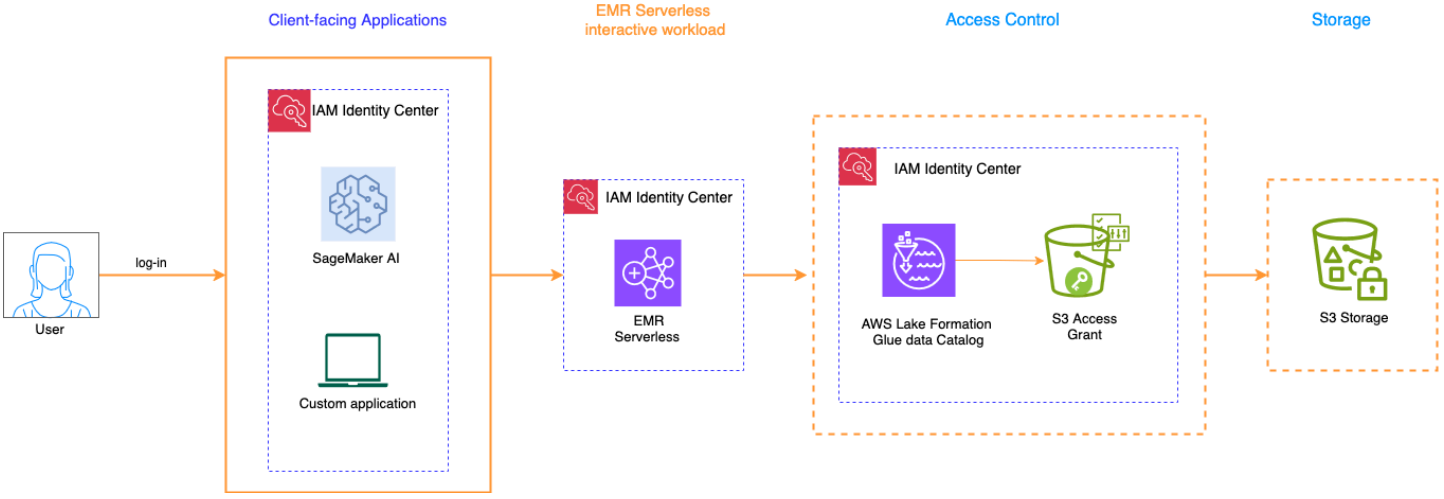
La propagation fiable des identités est une fonctionnalité du centre d'identité AWS IAM que les administrateurs des AWS services connectés peuvent utiliser pour accorder et auditer l'accès aux données de service. L'accès à ces données est basé sur les attributs utilisateur tels que les associations de groupe. La mise en place d'une propagation d'identité sécurisée nécessite une collaboration entre les administrateurs des AWS services connectés et les administrateurs de l'IAM Identity Center. Pour plus d'informations, reportez-vous à la section Conditions préalables et considérations du Guide de l'utilisateur d'IAM Identity Center.

Fonctionnalités et avantages

L'intégration du point de terminaison Apache Livy sans serveur EMR à IAM Identity Center Trusted Identity Center offre les avantages suivants :

- Possibilité d'appliquer l'autorisation au niveau des tables avec les identités d'Identity Center sur les tables du catalogue de données AWS Glue gérées par AWS Lake Formation.
- Vous avez la possibilité d'appliquer des autorisations avec les identités Identity Center sur les clusters Amazon Redshift.
- Permet le suivi de bout en bout des actions des utilisateurs à des fins d'audit.
- Vous avez la possibilité d'appliquer l'autorisation au niveau du préfixe Amazon S3 avec les identités Identity Center aux préfixes S3 gérés par S3 Access Grants.

Comment ça marche



Exemple de cas d’utilisation

Préparation des données et ingénierie des fonctionnalités

Les scientifiques des données de plusieurs équipes de recherche collaborent sur des projets complexes en s’aidant d’une plateforme de données unifiée. Ils se connectent à l' SageMaker IA à l'aide de leurs identifiants professionnels et ont immédiatement accès à un vaste lac de données partagé couvrant plusieurs AWS comptes. Alors qu'ils commencent à concevoir des fonctionnalités pour de nouveaux modèles d'apprentissage automatique, les sessions Spark lancées via EMR Serverless appliquent les politiques de sécurité de Lake Formation au niveau des colonnes et des lignes en fonction de leurs identités propagées. Les scientifiques peuvent préparer efficacement les données et concevoir des fonctionnalités à l'aide d'outils familiers, tandis que les équipes chargées de la conformité ont l'assurance que chaque interaction avec les données est automatiquement suivie et auditée. Cet environnement sécurisé et collaboratif accélère les pipelines de recherche tout en maintenant les normes strictes de protection des données requises dans les secteurs réglementés.

Commencer à utiliser Trusted-Identity Propagation

Cette section vous aide à configurer une application sans serveur EMR avec Apache Livy Endpoint pour l'intégrer à AWS IAM Identity Center et permettre la propagation d'identités fiables.

Conditions préalables

- Une instance de centre d'identité située dans la AWS région où vous souhaitez créer une propagation d'identité sécurisée a activé EMR Serverless Apache Livy Endpoint. Une instance d'Identity Center ne peut exister que dans une seule région pour un AWS compte. Reportez-vous à Activer le centre d'identité IAM et à fournir les utilisateurs et les groupes de votre source d'identités dans IAM Identity Center.
- Activez la propagation fiable des identités pour les services en aval tels que Lake Formation ou S3 Access Grants ou le cluster Amazon Redshift avec lequel une charge de travail interactive interagit pour accéder aux données.

Autorisations permettant de créer une application EMR sans serveur compatible avec la propagation d'identités fiables

Outre les autorisations de base requises pour accéder à EMR sans serveur, vous devez configurer des autorisations supplémentaires pour votre identité ou votre rôle IAM utilisé pour créer une

application EMR sans serveur compatible avec la propagation d'identités fiables. Pour la propagation d'identités fiables, EMR Serverless est creates/bootstraps un service unique qui gère l'application Identity Center dans votre compte, que le service utilise pour la validation de l'identité et la propagation de l'identité vers l'aval.

```
"sso:DescribeInstance",
"sso:CreateApplication",
"sso:DeleteApplication",
"sso:PutApplicationAuthenticationMethod",
"sso:PutApplicationAssignmentConfiguration",
"sso:PutApplicationGrant",
"sso:PutApplicationAccessScope"
```

- `sso:DescribeInstance`— Accorde l'autorisation de décrire et de valider l'InstanceArn IAM Identity Center que vous spécifiez en paramètre. identity-center-configuration
- `sso:CreateApplication`— Accorde l'autorisation de créer une application de centre d'identité IAM gérée sans serveur EMR qui est utilisée pour les actions. trusted-identity-propatgion
- `sso:DeleteApplication`— autorise le nettoyage d'une application IAM Identity Center gérée sans serveur EMR
- `sso:PutApplicationAuthenticationMethod`— Octroie l'autorisation d'installer AuthenticationMethod sur l'application IAM Identity Center gérée sans serveur EMR qui permet au principal du service sans serveur EMR d'interagir avec l'application IAM Identity Center.
- `sso:PutApplicationAssignmentConfiguration`— Accorde l'autorisation de définir le paramètre « U ser-assignment-not-required » sur l'application IAM Identity Center.
- `sso:PutApplicationGrant`— Autorise l'application des autorisations d'échange de jetons, IntrospectToken, RefreshToken et RevokeToken à une application IAM Identity Center.
- `sso:PutApplicationAccessScope`— Accorde l'autorisation d'appliquer à l'application IAM Identity Center une étendue en aval activée par la propagation des identités fiables. Nous appliquons les champs d'application « redshift:connect », « lakeformation:query » et « s3:read_write » pour activer ces services. trusted-identity-propagation

Création d'une application EMR sans serveur compatible avec la propagation d'identités fiables

Vous devez spécifier `–identity-center-configuration` le champ avec `identityCenterInstanceArn` pour activer la propagation des identités fiables dans l'application.

Utilisez l'exemple de commande suivant pour créer une application EMR sans serveur sur laquelle la propagation des identités fiables est activée.

> **Note**
>
> Vous devez également indiquer `--interactive-configuration '{"livyEndpointEnabled":true}'` que la propagation des identités fiables est activée pour Apache Livy Endpoint uniquement.

```
aws emr-serverless create-application \
  --release-label emr-7.8.0 \
  --type "SPARK" \
  --identity-center-configuration '{"identityCenterInstanceArn" :
 "arn:aws:sso:::instance/ssoins-123456789"}' \
  --interactive-configuration '{"livyEndpointEnabled":true}'
```

- `identity-center-configuration`— (facultatif) Active la propagation d'identités fiables par Identity Center si cela est spécifié.
- `identityCenterInstanceArn` (obligatoire) : ARN de l'instance Identity Center

Si vous ne disposez pas des autorisations requises pour le centre d'identité (mentionnées précédemment), créez d'abord l'application EMR sans serveur sans propagation d'identité fiable (par exemple, ne spécifiez pas de `–identity-center-configuration` paramètre), puis demandez à votre administrateur du centre d'identité d'activer la propagation de l'identité sécurisée en invoquant l'API de mise à jour de l'application, voir l'exemple ci-dessous :

```
aws emr-serverless update-application \
  --application-id applicationId \
  --identity-center-configuration '{"identityCenterInstanceArn" :
 "arn:aws:sso:::instance/ssoins-123456789"}'
```

EMR Serverless crée une application Identity Center gérée par service dans votre compte, que le service utilise pour les validations d'identité et la propagation des identités vers les services en aval. L'application Identity Center gérée créée par EMR Serverless est partagée entre toutes les applications trusted-identity-propagation EMR Serverless activées de votre compte.

> **Note**
>
> Ne modifiez pas manuellement les paramètres de l'application Identity Center gérée. Toute modification peut affecter toutes les applications EMR sans serveur trusted-identity-propagation activées dans votre compte.

Autorisations du rôle d'exécution des tâches pour propager l'identité

Comme EMR-Serverless utilise les job-execution-role informations d'identification améliorées pour propager l'identité aux services en aval AWS , la politique de confiance de Job Execution Role doit comporter des autorisations supplémentaires `sts:SetContext` pour améliorer les informations d'identification du rôle d'exécution des tâches par une identité afin de les autoriser à accéder aux trusted-identity-propagation services en aval, tels que S3 access-grant, Lake Formation ou Amazon Redshift. Pour en savoir plus sur la création d'un rôle, reportez-vous à la section Création d'un rôle d'exécution de tâches.

JSON

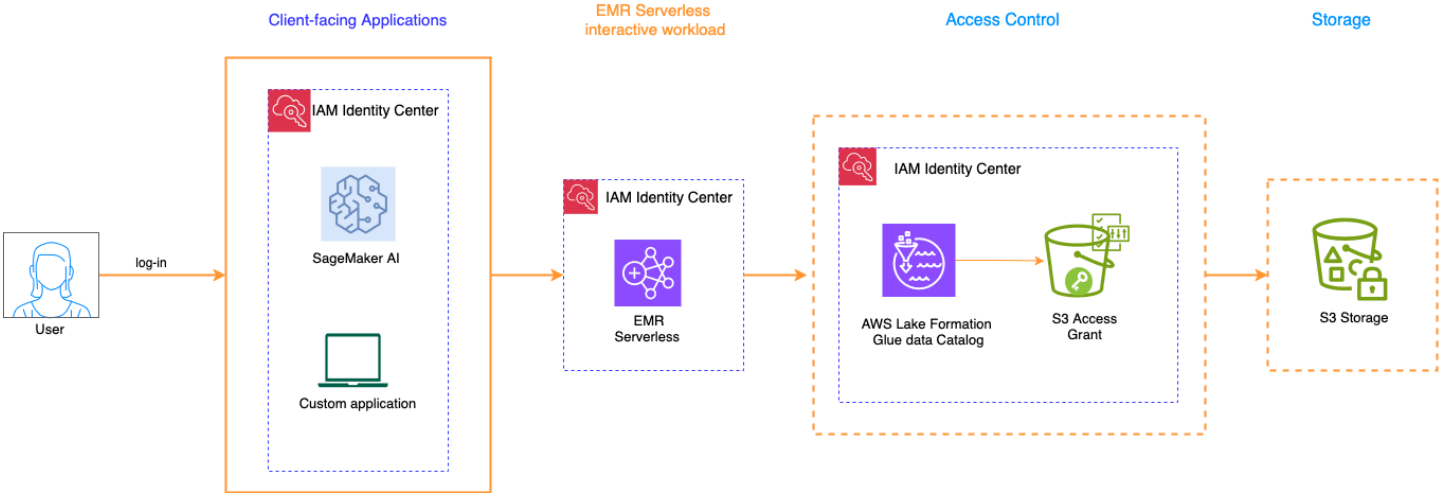
```
{
  "Version":"2012-10-17",
  "Statement": [
     {
     "Effect": "Allow",
     "Principal": {
     "Service": "emr-serverless.amazonaws.com"
     },
     "Action": [ "sts:AssumeRole", "sts:SetContext"]
     }
  ]
}
```

En outre, JobExecutionRole nécessite des autorisations pour les AWS services en aval que job-run invoquerait pour récupérer des données en utilisant l'identité de l'utilisateur. Reportez-vous aux liens ci-dessous pour configurer S3 Access Grant, Lake Formation.

- Utilisation de Lake Formation avec EMR sans serveur
- Utilisation des autorisations d'accès Amazon S3 avec EMR sans serveur

Propagation d'identité fiable pour les charges de travail interactives

Les étapes à suivre pour propager l'identité aux charges de travail interactives via un point de terminaison Apache Livy varient selon que vos utilisateurs interagissent avec un environnement de développement AWS géré tel que Amazon SageMaker AI ou avec votre propre environnement Notebook auto-hébergé en tant qu'application orientée client.



AWS environnement de développement géré

L'application AWS gérée orientée client suivante prend en charge la propagation d'identités fiables avec le point de terminaison Apache Livy sans serveur EMR :

- Amazon SageMaker AI

Environnement de bloc-notes auto-hébergé géré par le client

Pour activer la propagation d'identité fiable pour les utilisateurs d'applications développées sur mesure, consultez la section Accès aux AWS services par programmation à l'aide de la propagation d'identité fiable dans le blog sur la AWS sécurité.

Sessions d’arrière-plan pour les utilisateurs

Les sessions d'arrière-plan des utilisateurs permettent aux flux d'analyse et d'apprentissage automatique de longue durée de se poursuivre même après que l'utilisateur se soit déconnecté de l'interface de son bloc-notes. Cette fonctionnalité est mise en œuvre via l'intégration EMR Serverless avec la fonction de propagation d'identité fiable d'IAM Identity Center. Cette section explique les options de configuration et les comportements pour les sessions d'arrière-plan des utilisateurs.

> **Note**
>
> Les sessions d'arrière-plan des utilisateurs s'appliquent aux charges de travail Spark initiées via des interfaces de bloc-notes telles qu'Amazon SageMaker Unified Studio. L'activation ou la désactivation de cette fonctionnalité n'affecte que les nouvelles sessions Livy ; les sessions Livy actives existantes ne sont pas affectées.

Configurer les sessions d'arrière-plan des utilisateurs

Les sessions d'arrière-plan de l'utilisateur doivent être activées à deux niveaux pour fonctionner correctement :

1. Niveau d'instance IAM Identity Center : généralement configuré par les administrateurs iDC
2. Niveau d'application EMR Serverless : configuré par les administrateurs d'applications EMR Serverless

Activer les sessions utilisateur en arrière-plan pour les applications EMR sans serveur

Pour activer les sessions utilisateur en arrière-plan pour une application EMR sans serveur, vous devez définir le `userBackgroundSessionsEnabled` paramètre sur in `identityCenterConfiguration` lors de la création ou de la mise `true` à jour d'une application.

Conditions préalables

- Votre rôle IAM utilisé dans create/update l'application EMR Serverless doit disposer de cette autorisation. `sso:PutApplicationSessionConfiguration` Cette autorisation permet à EMR Serverless d'activer les sessions d'arrière-plan des utilisateurs au niveau de l'application iDC gérée par EMR Serverless.
- Votre application EMR Serverless doit utiliser l'étiquette de version 7.8 ou ultérieure et doit être activée sur Trusted-Identity Propagation.

Pour activer les sessions d'arrière-plan des utilisateurs à l'aide du AWS CLI

```
aws emr-serverless create-application \
    --name "my-analytics-app" \
    --type "SPARK" \
    --release-label "emr-7.8.0" \
```

```
    --identity-center-configuration '{"identityCenterInstanceArn":
"arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":
true}'
```

Pour mettre à jour une application existante :

```
aws emr-serverless update-application \
    --application-id applicationId \
    --identity-center-configuration '{"identityCenterInstanceArn":
"arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":
true}'
```

Matrice de configuration

La configuration effective de la session utilisateur en arrière-plan dépend à la fois du paramètre de l'application EMR Serverless et des paramètres au niveau de l'instance d'IAM Identity Center :

Matrice de configuration des sessions d'arrière-plan utilisateur

Centre userBackgroundSess ion d'identité IAM activé	EMR sans serveur activé userBackgroundSessions	Comportement
Oui	TRUE	Sessions d'arrière-plan utilisateur activées
Oui	FALSE	La session expire avec la déconnexion de l'utilisateur
Non	TRUE	L'application creation/update échoue avec une exception
Non	FALSE	La session expire avec la déconnexion de l'utilisateur

Durée par défaut de la session d’arrière-plan pour les utilisateurs

Par défaut, toutes les sessions d'arrière-plan des utilisateurs ont une durée maximale de 7 jours dans IAM Identity Center. Les administrateurs peuvent modifier cette durée dans la console IAM Identity Center. Ce paramètre s'applique au niveau de l'instance IAM Identity Center, affectant toutes les applications IAM Identity Center prises en charge au sein de cette instance.

- La durée peut être définie sur n'importe quelle valeur comprise entre 15 minutes et 90 jours.
- Ce paramètre est configuré dans la console IAM Identity Center sous Paramètres → Authentification → Configurer (section Tâches non interactives)

> **Note**
>
> Les sessions EMR Serverless Livy ont une durée maximale distincte de 24 heures. Les sessions se termineront lorsque la limite de session Livy ou la durée de la session d'arrière-plan de l'utilisateur sera atteinte, selon la première éventualité.

Impact de la désactivation des sessions d'arrière-plan des utilisateurs

Lorsque les sessions d'arrière-plan des utilisateurs sont désactivées dans IAM Identity Center :

Sessions Livy existantes

Continuez à fonctionner sans interruption s'ils ont été démarrés avec les sessions utilisateur en arrière-plan activées. Ces sessions continueront à utiliser leurs jetons de session d'arrière-plan existants jusqu'à ce qu'elles se terminent naturellement ou soient explicitement arrêtées.

Nouvelles sessions Livy

Utilisera le flux standard de propagation des identités fiables et s'arrêtera lorsque l'utilisateur se déconnectera ou que sa session interactive expirera (par exemple lors de la fermeture d'un JupyterLab bloc-notes Amazon SageMaker Unified Studio).

Modification de la durée des sessions d'arrière-plan des utilisateurs

Lorsque le paramètre de durée des sessions d'arrière-plan des utilisateurs est modifié dans IAM Identity Center :

Sessions Livy existantes

Continuez à s'exécuter avec la même durée de session en arrière-plan que celle avec laquelle ils ont été démarrés.

Nouvelles sessions Livy

Utilisera la nouvelle durée de session pour les sessions en arrière-plan.

Considérations

Conditions de fin de session

Lorsque vous utilisez des sessions utilisateur en arrière-plan, une session Livy continue de fonctionner jusqu'à ce que l'un des événements suivants se produise :

- La session d'arrière-plan de l'utilisateur expire (selon la configuration d'iDC, jusqu'à 90 jours)
- la session d’arrière-plan pour les utilisateurs est révoquée manuellement par un administrateur.
- La session Livy atteint son délai d'inactivité (par défaut : 1 heure après la dernière instruction exécutée)
- La session Livy atteint sa durée maximale (24 heures)
- L'utilisateur arrête ou redémarre explicitement le noyau du bloc-notes

Persistance des données

Lorsque vous utilisez des sessions d'arrière-plan utilisateur :

- Les utilisateurs ne peuvent pas se reconnecter à l'interface de leur bloc-notes pour voir les résultats une fois déconnectés
- Configurez vos instructions Spark pour écrire les résultats dans un stockage persistant (tel qu'Amazon S3) avant la fin de l'exécution

Incidences financières

- Les tâches continueront à être exécutées jusqu'à leur fin même une fois que les utilisateurs auront mis fin à leur JupyterLab session Amazon SageMaker Unified Studio et des frais seront facturés pendant toute la durée de l'exécution terminée.
- Surveillez vos sessions d'arrière-plan actives pour éviter les coûts inutiles liés à des sessions oubliées ou abandonnées.

Disponibilité des fonctions

Les sessions d'arrière-plan utilisateur pour EMR Serverless sont disponibles pour :

- Moteur Spark uniquement (le moteur Hive n'est pas pris en charge)

- Sessions interactives Livy uniquement (les tâches par lots et les tâches de streaming ne sont pas prises en charge)
- Libellés de version 7.8 et versions ultérieures d'EMR Serverless

Considérations relatives à l'intégration EMR sans serveur Trusted-Identity-Propagation

Tenez compte des points suivants lorsque vous utilisez IAM Identity Center Trusted-Identity-Propagation avec une application EMR Serverless :

- La propagation d'identités fiables via Identity Center est prise en charge sur Amazon EMR 7.8.0 et versions ultérieures, et uniquement avec Apache Spark.
- La propagation d'identité sécurisée ne peut être utilisée que pour les charges de travail interactives avec EMR Serverless via un point de terminaison Apache Livy. Les charges de travail interactives via EMR Studio ne prennent pas en charge la propagation d'identités fiables
- Les tâches par lots et les charges de travail en streaming ne prennent pas en charge la propagation fiable des identités
- Des contrôles d'accès précis utilisant AWS Lake Formation et utilisant Trusted Identity Propagation sont disponibles pour les charges de travail interactives avec EMR Serverless via un point de terminaison Apache Livy.
- La propagation d'identités fiables avec Amazon EMR est prise en charge dans les régions suivantes : AWS
  - af-south-1 – Afrique (Le Cap)
  - ap-east-1 – Asie-Pacifique (Hong Kong)
  - ap-northeast-1 – Asie-Pacifique (Tokyo)
  - ap-northeast-2, Asie-Pacifique (Séoul)
  - ap-northeast-3 – Asie-Pacifique (Osaka)
  - ap-south-1, Asie-Pacifique (Mumbai)
  - ap-southeast-1 – Asie-Pacifique (Singapour)
  - ap-southeast-2 – Asie-Pacifique (Sydney)
  - ap-southeast-3 – Asie-Pacifique (Jakarta)
  - ca-central-1, Canada (Centre)
  - ca-west-1 — Canada (Calgary)

- eu-central-1 – Europe (Francfort)
- eu-north-1, Europe (Stockholm)
- eu-south-1 – Europe (Milan)
- eu-south-2 — Europe (Espagne)
- eu-west-1 – Europe (Irlande)
- eu-west-2, Europe (Londres)
- eu-west-3, Europe (Paris)
- me-central-1 — Moyen-Orient (Émirats arabes unis)
- me-south-1 – Moyen-Orient (Bahreïn)
- sa-east-1, Amérique du Sud (São Paulo)
- us-east-1 – USA Est (Virginie du Nord)
- us-east-2 – USA Est (Ohio)
- us-west-1, USA Ouest (Californie du Nord)
- us-west-2 – USA Ouest (Oregon)

Utilisation de Lake Formation avec EMR sans serveur

Vous pouvez configurer les applications EMR Serverless pour utiliser Lake Formation avec un accès complet aux tables ou un contrôle d'accès détaillé. Pour plus de détails sur les fonctionnalités prises en charge dans chaque mode d'accès, consultez le tableau suivant.

Disponibilité des fonctions

Fonctionnalité	Disponible auprès de
Opérations de lecture (SELECT, DESCRIBE) pour les tables Hive, Iceberg	EMR 7,2 +
Vues multidialectales	EMR 7,6 +
Opérations de lecture (SELECT, DESCRIBE) pour les tables Delta Lake et Hudi	EMR 7,6 +
Accès complet aux tables pour Hive, Iceberg	EMR 7,9+

Fonctionnalité	Disponible auprès de
Accès complet aux tables pour Delta Lake	EMR 7,11 OU VERSION ULTÉRIEURE
Opérations d'écriture (DDL, DML) pour les tables Hive, Iceberg et Delta Lake	EMR 7,12 OU VERSION ULTÉRIEURE
Accès complet aux tables pour Hudi	EMR 7,12 OU VERSION ULTÉRIEURE

Accès complet aux tables de Lake Formation pour EMR Serverless

Avec les versions 7.8.0 et supérieures d'Amazon EMR, vous pouvez tirer parti de AWS Lake Formation with Glue Data Catalog, dans lequel le rôle d'exécution des tâches dispose d'autorisations complètes sur les tables, sans les limites d'un contrôle d'accès précis. Cette fonctionnalité vous permet de lire et d'écrire sur des tables protégées par Lake Formation à partir de votre lot EMR Serverless Spark et de vos tâches interactives. Consultez les sections suivantes pour en savoir plus sur Lake Formation et sur son utilisation avec EMR Serverless.

Utilisation de Lake Formation avec accès complet aux tables

Vous pouvez accéder aux tables du catalogue Glue Data protégées par AWS Lake Formation à partir de jobs EMR Serverless Spark ou de sessions interactives dans lesquelles le rôle d'exécution de la tâche dispose d'un accès complet aux tables. Il n'est pas nécessaire d'activer AWS Lake Formation sur l'application EMR Serverless. Lorsqu'une tâche Spark est configurée pour un accès complet aux tables (FTA), les informations d'identification de AWS Lake Formation sont utilisées pour les données read/write S3 des tables enregistrées par AWS Lake Formation, tandis que les informations d'identification du rôle d'exécution de la tâche seront utilisées pour read/write les tables non enregistrées auprès de AWS Lake Formation.

> **Important**
>
> N'activez pas AWS Lake Formation pour un contrôle d'accès précis. Une tâche ne peut pas exécuter simultanément un accès complet aux tables (FTA) et un contrôle d'accès détaillé (FGAC) sur le même cluster ou application EMR.

Étape 1 : Activer l'accès complet à la table dans Lake Formation

Pour utiliser le mode Full Table Access (FTA), vous devez autoriser les moteurs de requêtes tiers à accéder aux données sans la validation des balises de session IAM dans AWS Lake Formation. Pour l'activer, suivez les étapes de la section Intégration des applications pour un accès complet aux tables.

> **Note**
>
> Lorsque vous accédez à des tables entre comptes, l'accès complet aux tables doit être activé à la fois dans les comptes de producteurs et de consommateurs. De la même manière, lors de l'accès aux tables interrégionales, ce paramètre doit être activé à la fois dans les régions productrices et consommatrices.

Étape 2 : configurer des autorisations IAM pour le rôle d'exécution des tâches

Pour accéder en lecture ou en écriture aux données sous-jacentes, outre les autorisations de Lake Formation, un rôle d'exécution de tâche nécessite l'autorisation `lakeformation:GetDataAccess` IAM. Avec cette autorisation, Lake Formation accède à la demande d'informations d'identification temporaires pour accéder aux données.

Voici un exemple de politique expliquant comment fournir des autorisations IAM pour accéder à un script dans Amazon S3, télécharger des journaux vers S3, autoriser l'API AWS Glue et accéder à Lake Formation.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts"
      ]
```

```
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Étape 2.1 Configurer les autorisations de Lake Formation

- Les tâches Spark qui lisent des données depuis S3 nécessitent l'autorisation de Lake Formation SELECT.
- Les tâches Spark dont write/delete les données dans S3 nécessitent l'autorisation de Lake Formation ALL (SUPER).

- Les tâches Spark qui interagissent avec le catalogue Glue Data nécessitent les autorisations DESCRIBE, ALTER, DROP, selon le cas.

Pour plus d'informations, reportez-vous à la section Octroi d'autorisations sur les ressources du catalogue de données.

Étape 3 : Initialisation d'une session Spark pour un accès complet à la table à l'aide de Lake Formation

Conditions préalables

AWS Le catalogue de données Glue doit être configuré en tant que métastore pour accéder aux tables de Lake Formation.

Définissez les paramètres suivants pour configurer le catalogue Glue en tant que métastore :

```
--conf spark.sql.catalogImplementation=hive
--conf
 spark.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCata
```

Pour plus d'informations sur l'activation du catalogue de données pour EMR Serverless, reportez-vous à la section Configuration de Metastore pour EMR Serverless.

Pour accéder aux tables enregistrées auprès de AWS Lake Formation, les configurations suivantes doivent être définies lors de l'initialisation de Spark afin que Spark utilise les informations d'identification de AWS Lake Formation.

Hive

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
```

```
--conf spark.sql.catalog.spark_catalog.warehouse=S3_DATA_LOCATION
--conf spark.sql.catalog.spark_catalog.client.region=REGION
--conf spark.sql.catalog.spark_catalog.type=glue
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Delta Lake

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Hudi

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
--conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
--conf spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension
--conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.A` Configurez le système de fichiers EMR (EMRFS) ou EMR S3A pour utiliser les informations d'identification de AWS Lake Formation S3 pour les tables enregistrées par Lake Formation. Si la table n'est pas enregistrée, utilisez les informations d'identification du rôle d'exécution de la tâche.
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` et `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true` : configurez EMRFS pour utiliser l'en-tête de type de contenu application/x-directory au lieu du suffixe $folder$ lors de la création de dossiers S3. Cela est nécessaire lors de la lecture des tables de Lake Formation, car

les informations d'identification de Lake Formation ne permettent pas de lire les dossiers de tables portant le suffixe $folder$.

- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true` : configurez Spark pour ignorer la validation de l'emplacement vide de la table avant sa création. Cela est nécessaire pour les tables enregistrées dans Lake Formation, car les informations d'identification de Lake Formation permettant de vérifier l'emplacement vide ne sont disponibles qu'après la création de la table Glue Data Catalog. Sans cette configuration, les informations d'identification du rôle d'exécution de la tâche valideront l'emplacement de la table vide.
- `spark.sql.catalog.createDirectoryAfterTable.enabled=true` : configurez Spark pour créer le dossier Amazon S3 après la création de la table dans le métastore Hive. Cela est obligatoire pour les tables enregistrées dans Lake Formation, car les informations d'identification de Lake Formation permettant de créer le dossier S3 ne sont disponibles qu'après la création de la table Glue Data Catalog.
- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true`: configurez Spark pour qu'il supprime le dossier S3 avant la suppression de la table dans le métastore Hive. Cela est nécessaire pour les tables enregistrées par Lake Formation, car les informations d'identification de Lake Formation permettant de supprimer le dossier S3 ne sont pas disponibles après la suppression des tables du catalogue de données Glue.
- `spark.sql.catalog.<catalog>.glue.lakeformation-enabled=true`: Configurez le catalogue Iceberg pour utiliser les informations d'identification de AWS Lake Formation S3 pour les tables enregistrées par Lake Formation. Si la table n'est pas enregistrée, utilisez les informations d'identification d'environnement par défaut.

Configurer le mode d'accès complet aux tables dans SageMaker Unified Studio

Pour accéder aux tables enregistrées de Lake Formation à partir de sessions Spark interactives dans des JupyterLab blocs-notes, utilisez le mode d'autorisation de compatibilité. Utilisez la commande magique %%configure pour configurer votre configuration Spark. Choisissez la configuration en fonction de votre type de table :

For Hive tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
```

```
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

For Iceberg tables

```
%%configure -f
{
    "conf": {
        "spark.sql.catalog.spark_catalog":
 "org.apache.iceberg.spark.SparkSessionCatalog",
        "spark.sql.catalog.spark_catalog.warehouse": "S3_DATA_LOCATION",
        "spark.sql.catalog.spark_catalog.client.region": "REGION",
        "spark.sql.catalog.spark_catalog.type": "glue",
        "spark.sql.catalog.spark_catalog.glue.account-id": "ACCOUNT_ID",
        "spark.sql.catalog.spark_catalog.glue.lakeformation-enabled": "true",
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": "true",
    }
}
```

For Delta Lake tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

For Hudi tables

```
%%configure -f
{
```

```
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true,
        "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
        "spark.sql.extensions":
 "org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
        "spark.sql.catalog.spark_catalog":
 "org.apache.spark.sql.hudi.catalog.HoodieCatalog",
        "spark.serializer": "org.apache.spark.serializer.KryoSerializer"
    }
}
```

Remplacez les espaces réservés :

- `S3_DATA_LOCATION`: le chemin de votre compartiment S3
- `REGION`: AWS région (par exemple, us-east-1)
- `ACCOUNT_ID`: votre identifiant AWS de compte

> **Note**
>
> Vous devez définir ces configurations avant d'exécuter des opérations Spark dans votre bloc-notes.

Opérations prises en charge

Ces opérations utiliseront les informations d'identification de AWS Lake Formation pour accéder aux données de la table.

- CREATE TABLE
- ALTER TABLE
- INSERT INTO
- INSERT OVERWRITE

- UPDATE
- MERGE INTO
- DELETE FROM
- ANALYZE TABLE
- REPAIR TABLE
- DROP TABLE
- Requêtes de source de données Spark
- Écritures de source de données Spark

> **Note**
>
> Les opérations non répertoriées ci-dessus continueront à utiliser les autorisations IAM pour accéder aux données de la table.

Considérations

- Si une table Hive est créée à l'aide d'une tâche pour laquelle l'accès complet à la table n'est pas activé et qu'aucun enregistrement n'est inséré, les lectures ou écritures suivantes à partir d'une tâche avec accès complet à la table échoueront. Cela est dû au fait qu'EMR Spark sans accès complet à la table ajoute le `$folder$` suffixe au nom du dossier de la table. Pour résoudre cela, vous pouvez procéder comme suit :
  - Insérez au moins une ligne dans la table à partir d'une tâche pour laquelle FTA n'est pas activé.
  - Configurez la tâche pour laquelle FTA n'est pas activé pour ne pas utiliser de `$folder$` suffixe dans le nom du dossier dans S3. Cela peut être réalisé en configurant `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` pour Spark.
  - Créez un dossier S3 à l'emplacement de la table à `s3://path/to/table/table_name` l'aide de la console AWS S3 ou de la CLI AWS S3.
- L'accès complet aux tables est pris en charge avec le système de fichiers EMR (EMRFS) à partir de la version 7.8.0 d'Amazon EMR, et avec le système de fichiers S3A à partir de la version 7.10.0 d'Amazon EMR.
- L'accès complet aux tables est pris en charge pour les tables Hive, Iceberg, Delta et Hudi.
- Considérations relatives au support Hudi FTA Write :

  - Les écritures Hudi FTA doivent être utilisées HoodieCredentialedHadoopStorage pour la vente d'informations d'identification lors de l'exécution du travail. Définissez la configuration suivante lors de l'exécution des tâches Hudi : `hoodie.storage.class=org.apache.spark.sql.hudi.storage.HoodieCredentialedHadc`
  - La prise en charge de l'écriture Full Table Access (FTA) pour Hudi est disponible à partir de la version 7.12 d'Amazon EMR.
  - Le support d'écriture Hudi FTA ne fonctionne actuellement qu'avec les configurations Hudi par défaut. Les paramètres Hudi personnalisés ou autres que ceux par défaut peuvent ne pas être entièrement pris en charge et peuvent entraîner un comportement inattendu.
  - Le clustering pour les tables Hudi Merge-On-Read (MOR) n'est pas pris en charge à ce stade en mode d'écriture FTA.
- Les tâches faisant référence à des tables conformes aux règles FGAC (Lake Formation Fine-Grained Access Control) ou à Glue Data Catalog Views échoueront. Pour interroger une table avec des règles FGAC ou une vue du catalogue Glue Data, vous devez utiliser le mode FGAC. Vous pouvez activer le mode FGAC en suivant les étapes décrites dans la AWS documentation : Utilisation d'EMR Serverless with Lake AWS Formation pour un contrôle d'accès précis.
- L'accès complet aux tables ne prend pas en charge Spark Streaming.
- Lorsque vous écrivez Spark DataFrame dans une table Lake Formation, seul le mode APPEND est pris en charge pour les tables Hive et Iceberg : `df.write.mode("append").saveAsTable(table_name)`
- La création de tables externes nécessite des autorisations IAM.
- Comme Lake Formation met temporairement en cache les informations d'identification dans une tâche Spark, une tâche par lots Spark ou une session interactive en cours d'exécution risque de ne pas refléter les modifications d'autorisation.
- Vous devez utiliser un rôle défini par l'utilisateur et non un rôle lié à un service : Lake Formation requiert des rôles.

Hudi FTA Write Support - Opérations prises en charge

Le tableau suivant indique les opérations d'écriture prises en charge pour les tables Hudi Copy-On-Write (COW) et Merge-On-Read (MOR) en mode Accès complet aux tables :

Opérations d'écriture prises en charge par Hudi FTA

Type de table	Opération	Commande d'écriture SQL	Statut
VACHE	INSERT	INSÉRER DANS LE TABLEAU	Pris en charge
VACHE	INSERT	INSÉRER DANS LE TABLEAU - PARTITION (statique, dynamique)	Pris en charge
VACHE	INSERT	INSERT OVERWRITE	Pris en charge
VACHE	INSERT	INSERT OVERWRITE - PARTITION (statique, dynamique)	Pris en charge
UPDATE	UPDATE	TABLE DE MISE À JOUR	Pris en charge
VACHE	UPDATE	TABLE DE MISE À JOUR - Modifier la partition	Non pris en charge
DELETE	DELETE	SUPPRIMER DU TABLEAU	Pris en charge
ALTER	ALTER	MODIFIER LA TABLE - RENOMMER EN	Non pris en charge
VACHE	ALTER	MODIFIER LA TABLE -	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
		DÉFINIR LES PROPRIÉTÉS DE LA TABLE	
VACHE	ALTER	MODIFIER LA TABLE - DÉSACTIVER LES PROPRIÉTÉS DE LA TABLE	Pris en charge
VACHE	ALTER	MODIFIER LA TABLE - MODIFIER LA COLONNE	Pris en charge
VACHE	ALTER	MODIFIER LA TABLE - AJOUTER DES COLONNES	Pris en charge
VACHE	ALTER	MODIFIER LA TABLE - AJOUTER UNE PARTITION	Pris en charge
VACHE	ALTER	MODIFIER LA TABLE - SUPPRIMER LA PARTITION	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
VACHE	ALTER	MODIFIER LA TABLE - RÉCUPÉRER DES PARTITION S	Pris en charge
VACHE	ALTER	RÉPARER LES PARTITIONS DE SYNCHRONI SATION DES TABLES	Pris en charge
DROP	DROP	DROP TABLE	Pris en charge
VACHE	DROP	DROP TABLE - PURGE	Pris en charge
CREATE	CREATE	CRÉER UNE TABLE - Géré	Pris en charge
VACHE	CREATE	CRÉER UNE TABLE - PARTITIONNER PAR	Pris en charge
VACHE	CREATE	CRÉER UNE TABLE SI ELLE N'EXISTE PAS	Pris en charge
VACHE	CREATE	CREATE TABLE LIKE	Pris en charge
VACHE	CREATE	CREATE TABLE AS SELECT	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
CREATE	CREATE	CRÉER UNE TABLE avec EMPLACEMENT - Table externe	Non pris en charge
DATAFRAME (INSÉRER)	DATAFRAME (INSÉRER)	saveAsTable.Remplacer	Pris en charge
VACHE	DATAFRAME (INSÉRER)	saveAsTable.Ajouter	Non pris en charge
VACHE	DATAFRAME (INSÉRER)	saveAsTable.Ignorer	Pris en charge
VACHE	DATAFRAME (INSÉRER)	saveAsTable.ErrorIfExists	Pris en charge
VACHE	DATAFRAME (INSÉRER)	saveAsTable - Table externe (Path)	Non pris en charge
VACHE	DATAFRAME (INSÉRER)	enregistrer (chemin) - DF v1	Non pris en charge
MOR	INSERT	INSÉRER DANS LE TABLEAU	Pris en charge
MOR	INSERT	INSÉRER DANS LE TABLEAU - PARTITION (statique, dynamique)	Pris en charge
MOR	INSERT	INSERT OVERWRITE	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
MOR	INSERT	INSERT OVERWRITE - PARTITION (statique, dynamique)	Pris en charge
UPDATE	UPDATE	TABLE DE MISE À JOUR	Pris en charge
MOR	UPDATE	TABLE DE MISE À JOUR - Modifier la partition	Non pris en charge
DELETE	DELETE	SUPPRIMER DU TABLEAU	Pris en charge
ALTER	ALTER	MODIFIER LA TABLE - RENOMMER EN	Non pris en charge
MOR	ALTER	MODIFIER LA TABLE - DÉFINIR LES PROPRIÉTÉS DE LA TABLE	Pris en charge
MOR	ALTER	MODIFIER LA TABLE - DÉSACTIVE R LES PROPRIÉTÉS DE LA TABLE	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
MOR	ALTER	MODIFIER LA TABLE - MODIFIER LA COLONNE	Pris en charge
MOR	ALTER	MODIFIER LA TABLE - AJOUTER DES COLONNES	Pris en charge
MOR	ALTER	MODIFIER LA TABLE - AJOUTER UNE PARTITION	Pris en charge
MOR	ALTER	MODIFIER LA TABLE - SUPPRIMER LA PARTITION	Pris en charge
MOR	ALTER	MODIFIER LA TABLE - RÉCUPÉRER DES PARTITION S	Pris en charge
MOR	ALTER	RÉPARER LES PARTITIONS DE SYNCHRONI SATION DES TABLES	Pris en charge
DROP	DROP	DROP TABLE	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
MOR	DROP	DROP TABLE - PURGE	Pris en charge
CREATE	CREATE	CRÉER UNE TABLE - Géré	Pris en charge
MOR	CREATE	CRÉER UNE TABLE - PARTITIONNER PAR	Pris en charge
MOR	CREATE	CRÉER UNE TABLE SI ELLE N'EXISTE PAS	Pris en charge
MOR	CREATE	CREATE TABLE LIKE	Pris en charge
MOR	CREATE	CREATE TABLE AS SELECT	Pris en charge
CREATE	CREATE	CRÉER UNE TABLE avec EMPLACEMENT - Table externe	Non pris en charge
DATAFRAME (BOULEVERSÉ)	DATAFRAME (BOULEVERSÉ)	saveAsTable.Remplacer	Pris en charge
MOR	DATAFRAME (BOULEVERSÉ)	saveAsTable.Ajouter	Non pris en charge
MOR	DATAFRAME (BOULEVERSÉ)	saveAsTable.Ignorer	Pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
MOR	DATAFRAME (BOULEVERSÉ)	saveAsTable.ErrorIfExists	Pris en charge
MOR	DATAFRAME (BOULEVERSÉ)	saveAsTable - Table externe (Path)	Non pris en charge
MOR	DATAFRAME (BOULEVERSÉ)	enregistrer (chemin) - DF v1	Non pris en charge
DATAFRAME (SUPPRIMER)	DATAFRAME (SUPPRIMER)	saveAsTable.Ajouter	Non pris en charge
MOR	DATAFRAME (SUPPRIMER)	saveAsTable - Table externe (Path)	Non pris en charge
MOR	DATAFRAME (SUPPRIMER)	enregistrer (chemin) - DF v1	Non pris en charge
TRAME DE DONNÉES (BULK_INSERT)	TRAME DE DONNÉES (BULK_INSERT)	saveAsTable.Remplacer	Pris en charge
MOR	TRAME DE DONNÉES (BULK_INSERT)	saveAsTable.Ajouter	Non pris en charge
MOR	TRAME DE DONNÉES (BULK_INSERT)	saveAsTable.Ignorer	Pris en charge
MOR	TRAME DE DONNÉES (BULK_INSERT)	saveAsTable.ErrorIfExists	Pris en charge
MOR	TRAME DE DONNÉES (BULK_INSERT)	saveAsTable - Table externe (Path)	Non pris en charge

Type de table	Opération	Commande d'écriture SQL	Statut
MOR	TRAME DE DONNÉES (BULK_INSERT)	enregistrer (chemin) - DF v1	Non pris en charge

Utilisation d'EMR Serverless AWS Lake Formation pour un contrôle d'accès précis

Présentation de

Avec les versions 7.2.0 et supérieures d'Amazon EMR, vous pouvez AWS Lake Formation appliquer des contrôles d'accès précis aux tables du catalogue de données soutenues par S3. Cette fonctionnalité vous permet de configurer des contrôles d'accès au niveau des tables, des lignes, des colonnes et des cellules pour les read requêtes dans vos tâches Amazon EMR Serverless Spark. Pour configurer un contrôle d'accès précis pour les tâches par lots et les sessions interactives d'Apache Spark, utilisez EMR Studio. Consultez les sections suivantes pour en savoir plus sur Lake Formation et sur son utilisation avec EMR Serverless.

L'utilisation d'Amazon EMR Serverless AWS Lake Formation entraîne des frais supplémentaires. Pour plus d'informations, consultez la tarification d'Amazon EMR.

Comment fonctionne EMR Serverless avec AWS Lake Formation

L'utilisation d'EMR Serverless 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 lorsque EMR Serverless exécute des tâches. EMR Serverless 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 Lake Formation. Pour plus d'informations, reportez-vous à la section Qu'est-ce que c'est, considérations AWS Lake Formation et limites.

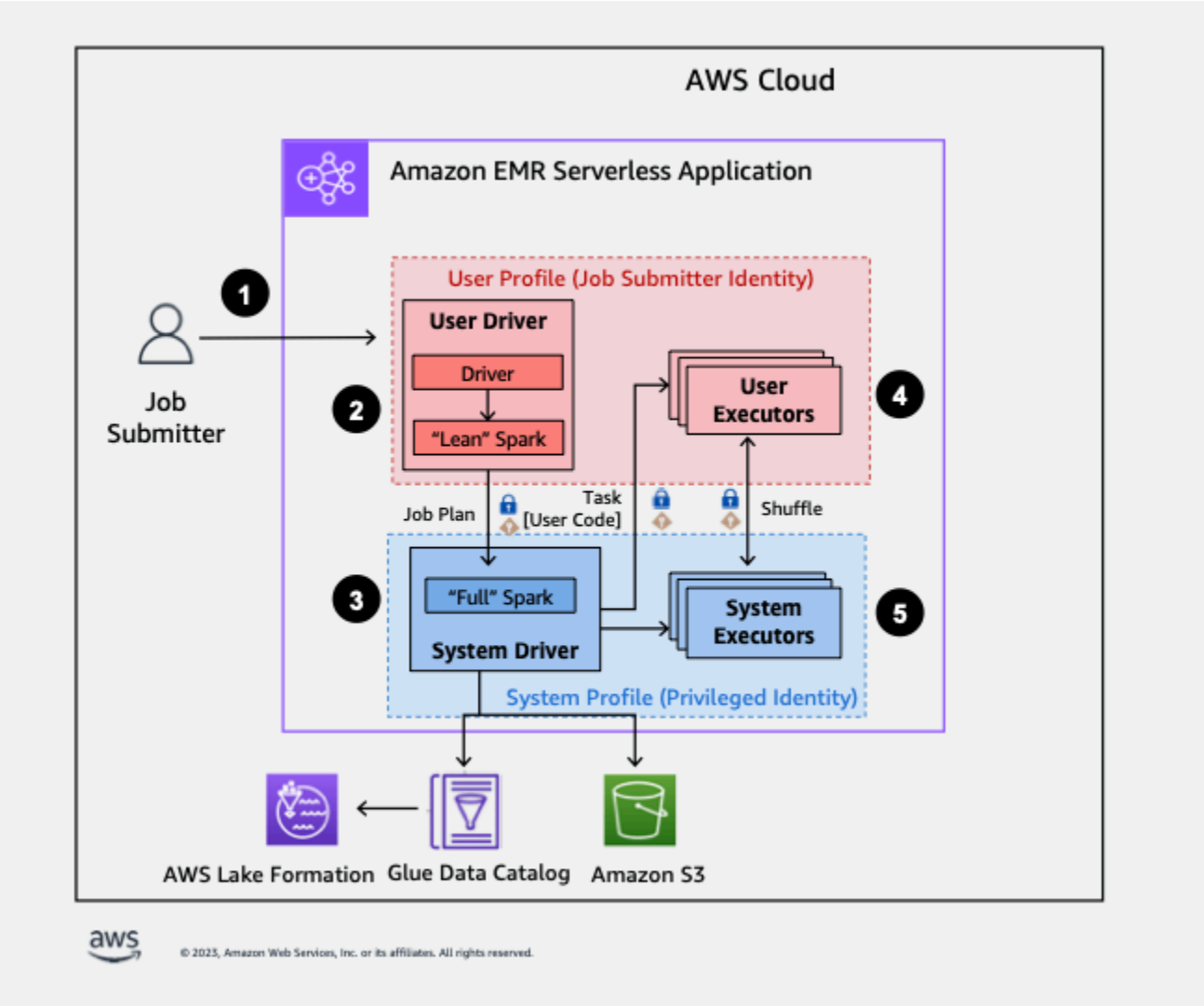
Lorsque vous utilisez une capacité préinitialisée avec Lake Formation, nous vous conseillons de disposer d'au moins deux pilotes Spark. Chaque tâche activée par Lake Formation utilise deux pilotes Spark, l'un pour le profil utilisateur et l'autre pour le profil système. Pour de meilleures performances, utilisez deux fois plus de conducteurs pour les tâches compatibles avec Lake Formation que si vous n'utilisez pas Lake Formation.

Lorsque vous exécutez des tâches Spark sur EMR Serverless, tenez également compte de l'impact de l'allocation dynamique sur la gestion des ressources et les performances du cluster. La configuration `spark.dynamicAllocation.maxExecutors` du nombre maximum d'exécuteurs par profil de ressource s'applique aux exécuteurs utilisateur et système. Si vous configurez ce nombre pour qu'il soit égal au nombre maximum autorisé d'exécuteurs, l'exécution de votre tâche risque de se bloquer car un type d'exécuteur utilise toutes les ressources disponibles, ce qui empêche l'autre exécuteur d'exécuter des tâches.

Pour ne pas manquer de ressources, EMR Serverless définit le nombre maximum d'exécuteurs par défaut par profil de ressource à 90 % de la valeur. `spark.dynamicAllocation.maxExecutors` Vous pouvez remplacer cette configuration lorsque vous spécifiez `spark.dynamicAllocation.maxExecutorsRatio` une valeur comprise entre 0 et 1. Configurez également les propriétés suivantes pour optimiser l'allocation des ressources et les performances globales :

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

Voici un aperçu général de la manière dont EMR Serverless accède aux données protégées par les politiques de sécurité de Lake Formation.



1. Un utilisateur soumet une tâche Spark à une application EMR sans serveur AWS Lake Formation compatible.
2. EMR Serverless envoie la tâche à un pilote utilisateur et l'exécute dans le profil utilisateur. Le pilote utilisateur exécute une version allégée de Spark qui n'est pas en mesure de lancer des tâches, de demander des exécuteurs, d'accéder à S3 ni au catalogue Glue. Il crée un plan de tâche.
3. EMR Serverless configure un deuxième pilote appelé pilote système et l'exécute dans le profil système (avec une identité privilégiée). EMR Serverless configure un canal TLS crypté entre les deux pilotes pour la communication. Le pilote utilisateur utilise le canal pour envoyer les plans de tâche au pilote système. Le pilote système n'exécute pas le code soumis par l'utilisateur. Il exécute Spark dans son intégralité et communique avec S3 et le catalogue de données pour

l'accès aux données. Il demande des exécuteurs et compile le plan de tâche en une séquence d'étapes d'exécution.

4. EMR Serverless exécute ensuite les étapes sur les exécuteurs à l'aide du pilote utilisateur ou du pilote système. À n'importe quelle étape, 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 des filtres de sécurité AWS Lake Formation ou qui appliquent des filtres de sécurité sont déléguées aux exécuteurs du système.

Favoriser la formation de Lake dans Amazon EMR

Pour activer Lake Formation, définissez le paramètre de configuration `spark.emr-serverless.lakeformation.enabled` d'exécution sur « `true` `spark-defaults` sous-classification » lors de la création d'une application EMR Serverless.

```
aws emr-serverless create-application \
    --release-label emr-7.12.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

Vous pouvez également activer Lake Formation lorsque vous créez une nouvelle application dans EMR Studio. Choisissez Utiliser Lake Formation pour un contrôle d'accès précis, disponible dans la section Configurations supplémentaires.

Le chiffrement entre utilisateurs est activé par défaut lorsque vous utilisez Lake Formation avec EMR Serverless. Il n'est donc pas nécessaire de réactiver explicitement le chiffrement entre utilisateurs.

Activer Lake Formation pour les emplois de Spark

Pour activer Lake Formation pour des tâches Spark individuelles, définissez ce paramètre `spark.emr-serverless.lakeformation.enabled` sur true lors de l'utilisation`spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

Autorisations IAM du rôle d’exécution des tâches

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 sur l’opération d’API `glue:Get*`.

Voici un exemple de politique expliquant comment fournir les autorisations IAM pour accéder à un script dans Amazon S3, le chargement de journaux sur S3, les autorisations d’API AWS Glue et les autorisations d’accès à Lake Formation.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts",
        "arn:aws:s3:::*.amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
```

```
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Configuration des autorisations de Lake Formation pour le rôle d'exécution des tâches

Tout d'abord, enregistrez l'emplacement de votre table Hive avec Lake Formation. Créez ensuite des autorisations pour votre rôle d'exécution des tâches dans la table de votre choix. Pour plus de détails sur la Lake Formation, reportez-vous à Qu'est-ce que c'est AWS Lake Formation ? dans le Guide AWS Lake Formation du développeur.

Après avoir configuré les autorisations de Lake Formation, soumettez des tâches Spark sur Amazon EMR Serverless. Pour plus d'informations sur les tâches Spark, reportez-vous aux exemples de Spark.

Soumission d'une exécution de tâche

Une fois que vous aurez fini de configurer les subventions Lake Formation, vous pourrez soumettre des tâches Spark sur EMR Serverless. La section qui suit présente des exemples de configuration et de soumission des propriétés d'exécution des tâches.

Conditions d'autorisation

Tables non enregistrées dans AWS Lake Formation

Pour les tables non enregistrées auprès de Amazon S3 AWS Lake Formation, le rôle d'exécution des tâches accède à la fois au catalogue de données AWS Glue et aux données des tables sous-jacentes. Cela nécessite que le rôle d'exécution des tâches dispose des autorisations IAM appropriées pour les opérations AWS Glue et Amazon S3.

Tables enregistrées dans AWS Lake Formation

Pour les tables enregistrées auprès de AWS Lake Formation, le rôle d'exécution du job accède aux métadonnées du catalogue de données AWS Glue, tandis que les informations d'identification temporaires fournies par Lake Formation accèdent aux données des tables sous-jacentes dans Amazon S3. Les autorisations de Lake Formation requises pour exécuter une opération dépendent du catalogue de données AWS Glue et des appels d'API Amazon S3 lancés par la tâche Spark et peuvent être résumées comme suit :

- L'autorisation DESCRIBE permet au rôle d'exécution de lire les métadonnées d'une table ou d'une base de données dans le catalogue de données
- L'autorisation ALTER permet au rôle d'exécution de modifier les métadonnées de table ou de base de données dans le catalogue de données

- L'autorisation DROP permet au rôle d'exécution de supprimer les métadonnées de table ou de base de données du catalogue de données
- L'autorisation SELECT permet au rôle d'exécution de lire les données des tables depuis Amazon S3
- L'autorisation INSERT permet au rôle d'exécution d'écrire des données de table sur Amazon S3
- L'autorisation DELETE permet au rôle d'exécution de supprimer les données de table d'Amazon S3

> **Note**
>
> Lake Formation évalue les autorisations de manière paresseuse lorsqu'une tâche Spark appelle AWS Glue pour récupérer les métadonnées des tables et Amazon S3 pour récupérer les données des tables. Les tâches qui utilisent un rôle d'exécution avec des autorisations insuffisantes n'échoueront pas tant que Spark n'aura pas AWS passé un appel à Glue ou Amazon S3 nécessitant l'autorisation manquante.

> **Note**
>
> Dans la matrice de tableau prise en charge suivante :
>
> - Les opérations marquées comme Supported utilisent exclusivement les informations d'identification de Lake Formation pour accéder aux données des tables enregistrées auprès de Lake Formation. Si les autorisations de Lake Formation sont insuffisantes, l'opération ne se basera pas sur les informations d'identification du rôle d'exécution. Pour les tables non enregistrées auprès de Lake Formation, les informations d'identification du rôle d'exécution du travail accèdent aux données de la table.
>
> - Les opérations marquées comme étant prises en charge par des autorisations IAM sur le site Amazon S3 n'utilisent pas les informations d'identification de Lake Formation pour accéder aux données des tables sous-jacentes dans Amazon S3. Pour exécuter ces opérations, le rôle d'exécution des tâches doit disposer des autorisations Amazon S3 IAM nécessaires pour accéder aux données de la table, que la table soit enregistrée ou non auprès de Lake Formation.

Hive

Opération	AWS Lake Formation autorisations	État du support
SELECT	SELECT	Pris en charge
CREATE TABLE	CRÉER_TABLE	Pris en charge
CREATE TABLE LIKE	CRÉER_TABLE	Compatible avec les autorisations IAM sur le site Amazon S3
CREATE TABLE AS SELECT	CRÉER_TABLE	Compatible avec les autorisations IAM sur le site Amazon S3
DESCRIBE TABLE	DESCRIBE	Pris en charge
SHOW TBLPROPERTIES	DESCRIBE	Pris en charge

Opération	AWS Lake Formation autorisations	État du support
SHOW COLUMNS	DESCRIBE	Pris en charge
SHOW PARTITIONS	DESCRIBE	Pris en charge
SHOW CREATE TABLE	DESCRIBE	Pris en charge
MODIFIER LE TABLEAU `tablename`	SELECT et ALTER	Pris en charge
MODIFIER L'EMPLACEMENT DU `tablename` SET DE TABLES	-	Non pris en charge
MODIFIER LA TABLE `tablename` AJOUTER UNE PARTITION	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
REPAIR TABLE	SELECT et ALTER	Pris en charge
CHARGER DES DONNÉES		Non pris en charge
INSERT	INSÉRER et MODIFIER	Pris en charge
INSERT OVERWRITE	SÉLECTIONNER, INSÉRER, SUPPRIMER et MODIFIER	Pris en charge
DROP TABLE	SÉLECTIONNER, SUPPRIMER, SUPPRIMER et MODIFIER	Pris en charge
TRUNCATE TABLE	SÉLECTIONNER, INSÉRER, SUPPRIMER et MODIFIER	Pris en charge

Opération	AWS Lake Formation autorisations	État du support
Dataframe Writer V1	Identique à l'opération SQL correspondante	Pris en charge lors de l'ajout de données à une table existante. Reportez-vous aux considérations et limites pour plus d'informations
Dataframe Writer V2	Identique à l'opération SQL correspondante	Pris en charge lors de l'ajout de données à une table existante. Reportez-vous aux considérations et limites pour plus d'informations

Iceberg

Opération	AWS Lake Formation autorisations	État du support
SELECT	SELECT	Pris en charge
CREATE TABLE	CRÉER_TABLE	Pris en charge
CREATE TABLE LIKE	CRÉER_TABLE	Compatible avec les autorisations IAM sur le site Amazon S3
CREATE TABLE AS SELECT	CRÉER_TABLE	Compatible avec les autorisations IAM sur le site Amazon S3
REMPLACER LE TABLEAU EN TANT QUE SELECT	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
DESCRIBE TABLE	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW TBLPROPERTIES	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3

Opération	AWS Lake Formation autorisations	État du support
SHOW CREATE TABLE	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
ALTER TABLE	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
ALTER TABLE SET LOCATION	SÉLECTIONNER, INSÉRER et MODIFIER	Compatible avec les autorisations IAM sur le site Amazon S3
MODIFIER L'ÉCRITUR E D'UNE TABLE ORDONNÉE PAR	SÉLECTIONNER, INSÉRER et MODIFIER	Compatible avec les autorisations IAM sur le site Amazon S3
MODIFIER L'ÉCRITUR E D'UNE TABLE DISTRIBUÉE PAR	SÉLECTIONNER, INSÉRER et MODIFIER	Compatible avec les autorisations IAM sur le site Amazon S3
MODIFIER LE TABLEAU RENOMMER LE TABLEAU	CREATE_TABLE et DROP	Pris en charge
INSERT INTO	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
INSERT OVERWRITE	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
DELETE	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
UPDATE	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
MERGE INTO	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge

Opération	AWS Lake Formation autorisations	État du support
DROP TABLE	SÉLECTIONNEZ, SUPPRIMEZ et SUPPRIMEZ	Pris en charge
DataFrame Writer V1	-	Non pris en charge
DataFrame Writer V2	Identique à l'opération SQL correspondante	Pris en charge lors de l'ajout de données à une table existante. Reportez-vous à la section Considérations et limites pour plus d'informations.
Tables de métadonnées	SELECT	Pris en charge. Certaines tables sont masquées. Reportez-vous à la section Considérations et limites pour plus d'informations.
Procédures stockées	-	Pris en charge pour les tables répondant aux conditions suivantes : • Tables non enregistrées dans AWS Lake Formation • Tables qui n'utilisent pas `register_table` et `migrate` Reportez-vous à la section Considérations et limites pour plus d'informations.

Configuration de Spark pour Iceberg : l'exemple suivant montre comment configurer Spark avec Iceberg. Pour exécuter des tâches Iceberg, saisissez les `spark-submit` propriétés suivantes.

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
```

```
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

Hudi

Opération	AWS Lake Formation autorisations	État du support
SELECT	SELECT	Pris en charge
CREATE TABLE	CRÉER_TABLE	Compatible avec les autorisations IAM sur le site Amazon S3
CREATE TABLE LIKE	CRÉER_TABLE	Compatible avec les autorisations IAM sur le site Amazon S3
CREATE TABLE AS SELECT	-	Non pris en charge
DESCRIBE TABLE	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW TBLPROPERTIES	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW COLUMNS	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW CREATE TABLE	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3

Opération	AWS Lake Formation autorisations	État du support
ALTER TABLE	SELECT	Compatible avec les autorisations IAM sur le site Amazon S3
INSERT INTO	SELECT et ALTER	Compatible avec les autorisations IAM sur le site Amazon S3
INSERT OVERWRITE	SELECT et ALTER	Compatible avec les autorisations IAM sur le site Amazon S3
DELETE	-	Non pris en charge
UPDATE	-	Non pris en charge
MERGE INTO	-	Non pris en charge
DROP TABLE	SELECT et DROP	Compatible avec les autorisations IAM sur le site Amazon S3
DataFrame Writer V1	-	Non pris en charge
DataFrame Writer V2	Identique à l'opération SQL correspondante	Compatible avec les autorisations IAM sur le site Amazon S3
Tables de métadonnées	-	Non pris en charge
Maintenance des tables et fonctionnalités utilitaires	-	Non pris en charge

Les exemples suivants configurent Spark avec Hudi, en spécifiant l'emplacement des fichiers et les autres propriétés nécessaires à son utilisation.

Configuration Spark pour Hudi : lorsqu'il est utilisé dans un bloc-notes, cet extrait indique le chemin d'accès au fichier JAR du bundle Hudi Spark, qui active la fonctionnalité Hudi dans Spark. Il configure également Spark pour utiliser le catalogue de données AWS Glue comme métastore.

```
%%configure -f
{
    "conf": {
        "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
        "spark.hadoop.hive.metastore.client.factory.class":
 "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
        "spark.serializer": "org.apache.spark.serializer.JavaSerializer",
        "spark.sql.catalog.spark_catalog":
 "org.apache.spark.sql.hudi.catalog.HoodieCatalog",
        "spark.sql.extensions":
 "org.apache.spark.sql.hudi.HoodieSparkSessionExtension"
    }
}
```

Configuration Spark pour Hudi with AWS Glue : cet extrait, lorsqu'il est utilisé dans un bloc-notes, active Hudi en tant que format de lac de données pris en charge et garantit la disponibilité des bibliothèques et des dépendances Hudi.

```
%%configure
{
    "--conf": "spark.serializer=org.apache.spark.serializer.JavaSerializer --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog --
conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
    "--datalake-formats": "hudi",
    "--enable-glue-datacatalog": True,
    "--enable-lakeformation-fine-grained-access": "true"
}
```

Delta Lake

Opération	AWS Lake Formation autorisations	État du support
SELECT	SELECT	Pris en charge
CREATE TABLE	CRÉER_TABLE	Pris en charge

Opération	AWS Lake Formation autorisations	État du support
CREATE TABLE LIKE	-	Non pris en charge
CREATE TABLE AS SELECT	CRÉER_TABLE	Pris en charge
REMPLACER LE TABLEAU EN TANT QUE SELECT	SÉLECTIONNER, INSÉRER et MODIFIER	Pris en charge
DESCRIBE TABLE	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW TBLPROPERTIES	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW COLUMNS	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
SHOW CREATE TABLE	DESCRIBE	Compatible avec les autorisations IAM sur le site Amazon S3
ALTER TABLE	SÉLECTIONNER et INSÉRER	Pris en charge
ALTER TABLE SET LOCATION	SÉLECTIONNER et INSÉRER	Compatible avec les autorisations IAM sur le site Amazon S3
MODIFIER LE `tablename` CLUSTER DE TABLES PAR	SÉLECTIONNER et INSÉRER	Compatible avec les autorisations IAM sur le site Amazon S3

Opération	AWS Lake Formation autorisations	État du support
MODIFIER LE TABLEAU `tablename` AJOUTER UNE CONTRAINTE	SÉLECTIONNER et INSÉRER	Compatible avec les autorisations IAM sur le site Amazon S3
MODIFIER LA CONTRAINTE DE `tablename` SUPPRESSION DE LA TABLE	SÉLECTIONNER et INSÉRER	Compatible avec les autorisations IAM sur le site Amazon S3
INSERT INTO	SÉLECTIONNER et INSÉRER	Pris en charge
INSERT OVERWRITE	SÉLECTIONNER et INSÉRER	Pris en charge
DELETE	SÉLECTIONNER et INSÉRER	Pris en charge
UPDATE	SÉLECTIONNER et INSÉRER	Pris en charge
MERGE INTO	SÉLECTIONNER et INSÉRER	Pris en charge
DROP TABLE	SÉLECTIONNER, SUPPRIMER et SUPPRIMER	Pris en charge
DataFrame Writer V1	-	Non pris en charge
DataFrame Writer V2	Identique à l'opération SQL correspondante	Pris en charge
Maintenance des tables et fonctionnalités utilitaires	-	Non pris en charge

EMR sans serveur avec Delta Lake : pour utiliser Delta Lake with Lake Formation sur EMR Serverless, exécutez la commande suivante :

```
spark-sql \
  --conf
 spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserver
 \
  --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \
```

Débogage de tâches

> **Note**
>
> Grâce à cette fonctionnalité, accédez aux journaux stdout et stderr pour les responsables du profil du système qui peuvent contenir des informations sensibles et non filtrées. L'autorisation suivante ne doit être utilisée que pour accéder à des données non liées à la production. Pour les applications créées pour être utilisées avec des tâches de production, nous vous conseillons vivement d'attribuer ces autorisations uniquement aux administrateurs ou aux utilisateurs disposant d'un accès élevé aux données.

Avec EMR-7.3.0 et versions ultérieures, EMR Serverless permet d'activer la fonctionnalité d'auto-débogage pour les tâches par lots compatibles avec Lake Formation. Pour ce faire, utilisez le nouveau paramètre accessSystemProfileLogs dans l'GetDashboardForJobRunAPI. Si accessSystemProfileLogs est défini sur true, vous pouvez accéder aux journaux stdout et stderr pour les gestionnaires de profils système, qui peuvent être utilisés pour déboguer une tâche par lots EMR sans serveur compatible avec Lake Formation.

```
aws emr-serverless get-dashboard-for-job-run \
  --application-id application-id
  --job-run-id job-run-id
  --access-system-profile-logs
```

Autorisations requises

Le principal qui souhaite déboguer des tâches par lots compatibles avec Lake Formation à l'aide de Lake Formation GetDashboardForJobRundoit disposer des autorisations supplémentaires suivantes :

```
{
    "Sid": "AccessSystemProfileLogs",
    "Effect": "Allow",
    "Action": [
        "emr-serverless:GetDashboardForJobRun",
        "emr-serverless:AccessSystemProfileLogs",
        "glue:GetDatabases",
        "glue:SearchTables"
    ],
    "Resource": [
        "arn:aws:emr-serverless:region:account-id:/applications/applicationId/
jobruns/jobid",
        "arn:aws:glue:region:account-id:catalog",
        "arn:aws:glue:region:account-id:database/*",
        "arn:aws:glue:region:account-id:table/*/*"
    ]
}
```

Considérations

Les journaux de profil système destinés au débogage sont visibles pour les tâches qui accèdent aux bases de données ou aux tables de Lake Formation avec le même compte que la tâche. Ils ne sont pas visibles dans les scénarios suivants :

- Si le catalogue de données géré à l'aide des autorisations de Lake Formation comporte des bases de données et des tables entre comptes
- Si le catalogue de données géré à l'aide des autorisations de Lake Formation comporte des liens vers des ressources

Utilisation des vues du catalogue de données Glue

Vous pouvez créer et gérer des vues dans le catalogue de données AWS Glue pour les utiliser avec EMR Serverless. Ces vues sont communément appelées vues AWS Glue Data Catalog. Ces vues sont utiles car elles prennent en charge plusieurs moteurs de requêtes SQL, ce qui vous permet d'accéder à la même vue sur différents AWS services, tels que EMR Serverless et Amazon Amazon Athena Redshift.

En créant une vue dans le catalogue de données, utilisez des autorisations de ressources et des contrôles d'accès basés sur des balises AWS Lake Formation pour autoriser l'accès à celle-ci. Avec cette méthode de contrôle d'accès, il n'est pas nécessaire de configurer un accès supplémentaire aux

tables que vous avez référencées lors de la création de la vue. Cette méthode d'octroi d'autorisations est appelée sémantique du définisseur, et ces vues sont appelées vues du définisseur. Pour plus d'informations sur le contrôle d'accès à Lake Formation, consultez la section Octroi et révocation d'autorisations sur les ressources du catalogue de données dans le guide du développeur de AWS Lake Formation.

Les vues du catalogue de données sont utiles dans les cas d'utilisation suivants :

- Contrôle précis des accès : vous pouvez créer une vue qui restreint l'accès aux données en fonction des autorisations dont l'utilisateur a besoin. Par exemple, vous pouvez utiliser des vues du catalogue de données pour empêcher les employés qui ne travaillent pas dans le service des ressources humaines (RH) de voir des données d'identification personnelle (PII).
- Définition complète de la vue : en appliquant des filtres à votre vue dans le catalogue de données, vous vous assurez que les enregistrements de données disponibles dans une vue du catalogue de données sont toujours complets.
- Sécurité renforcée — La définition de requête utilisée pour créer la vue doit être complète. Cet avantage signifie que les vues du catalogue de données sont moins sensibles aux commandes SQL d'acteurs malveillants.
- Partage de données simplifié : partagez des données avec d'autres AWS comptes sans déplacer de données. Pour plus d'informations, reportez-vous à la section Partage de données entre comptes dans Lake Formation.

Création d'un affichage du Catalogue de données

Il existe différentes manières de créer une vue du catalogue de données. Il s'agit notamment d'utiliser le AWS CLI ou Spark SQL. Voici quelques exemples.

Using SQL

Ce qui suit illustre la syntaxe permettant de créer une vue du catalogue de données. Notez le type de `MULTI DIALECT` vue. Cela distingue la vue du catalogue de données des autres vues. Le `SECURITY` prédicat est spécifié sous la forme`DEFINER`. Cela indique une vue du catalogue de données avec `DEFINER` sémantique.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment], ...) ]
[ COMMENT view_comment ]
[TBLPROPERTIES (property_name = property_value, ... )]
SECURITY DEFINER
```

```
AS query;
```

Voici un exemple d'CREATEinstruction dont la syntaxe est conforme à la syntaxe :

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY order_date
```

Vous pouvez également créer une vue en mode d'exécution à sec, à l'aide de SQL, pour tester la création de vues, sans créer réellement la ressource. L'utilisation de cette option entraîne un « essai à sec » qui valide l'entrée et, si la validation aboutit, renvoie le JSON de l'objet de table AWS Glue qui représentera la vue. Dans ce cas, la vue réelle n'est pas créée.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
SECURITY DEFINER
[ SHOW VIEW JSON ]
AS view-sql
```

Using the AWS CLI

> **Note**
>
> Lorsque vous utilisez la commande CLI, le code SQL utilisé pour créer la vue n'est pas analysé. Cela peut entraîner le cas où la vue est créée, mais les requêtes échouent. Assurez-vous de tester votre syntaxe SQL avant de créer la vue.

Vous utilisez la commande CLI suivante pour créer une vue :

```
aws glue create-table --cli-input-json '{
  "DatabaseName": "database",
  "TableInput": {
    "Name": "view",
    "StorageDescriptor": {
      "Columns": [
        {
          "Name": "col1",
          "Type": "data-type"
```

```
        },
        ...
        {
          "Name": "col_n",
          "Type": "data-type"
        }
      ],
      "SerdeInfo": {}
    },
    "ViewDefinition": {
      "SubObjects": [
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-table1",
        ...
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-tableN",
       ],
      "IsProtected": true,
      "Representations": [
        {
          "Dialect": "SPARK",
          "DialectVersion": "1.0",
          "ViewOriginalText": "Spark-SQL",
          "ViewExpandedText": "Spark-SQL"
        }
      ]
    }
  }
}'
```

Opérations de vue prises en charge

Les fragments de commande suivants vous montrent différentes manières d'utiliser les vues du catalogue de données :

- CRÉER UNE VUE

  Crée une vue du catalogue de données. Voici un exemple qui montre comment créer une vue à partir d'une table existante :

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table
```

- MODIFIER LA VUE

Syntaxe disponible :

- `ALTER VIEW view_name [FORCE] ADD DIALECT AS query`
- `ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query`
- `ALTER VIEW view_name DROP DIALECT`

Vous pouvez utiliser l'option `FORCE ADD DIALECT` pour forcer la mise à jour du schéma et des sous-objets conformément au nouveau dialecte du moteur. Notez que cela peut entraîner des erreurs de requête si vous n'utilisez pas également `FORCE` pour mettre à jour d'autres dialectes du moteur. Voici un exemple :

```
ALTER VIEW catalog_view FORCE ADD DIALECT
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY orderdate;
```

Voici comment modifier une vue pour mettre à jour le dialecte :

```
ALTER VIEW catalog_view UPDATE DIALECT AS
SELECT count(*) FROM my_catalog.my_database.source_table;
```

- DÉCRIRE LA VUE

  Syntaxe disponible pour décrire une vue :

  - `SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]`— Si l'utilisateur dispose des autorisations AWS Glue and Lake Formation requises pour décrire la vue, il peut répertorier les colonnes. Voici quelques exemples de commandes permettant d'afficher des colonnes :

    ```
    SHOW COLUMNS FROM my_database.source_table;
    SHOW COLUMNS IN my_database.source_table;
    ```

  - `DESCRIBE view_name`— Si l'utilisateur dispose des autorisations AWS Glue and Lake Formation requises pour décrire la vue, il peut répertorier les colonnes de la vue ainsi que ses métadonnées.

- SUPPRIMER LA VUE

  Syntaxe disponible :

- `DROP VIEW [ IF EXISTS ] view_name`

  L'exemple suivant montre une instruction DROP qui teste si une vue existe avant de la supprimer :

```
DROP VIEW IF EXISTS catalog_view;
```

- AFFICHER CRÉER UNE VUE
  - `SHOW CREATE VIEW view_name` : affiche l'instruction SQL qui crée la vue spécifiée. Voici un exemple qui montre comment créer une vue du catalogue de données :

```
SHOW CREATE TABLE my_database.catalog_view;
CREATE PROTECTED MULTI DIALECT VIEW my_catalog.my_database.catalog_view (
  net_profit,
  customer_id,
  item_id,
  sold_date)
TBLPROPERTIES (
  'transient_lastDdlTime' = '1736267222')
SECURITY DEFINER AS SELECT * FROM
my_database.store_sales_partitioned_lf WHERE customer_id IN (SELECT customer_id
 from source_table limit 10)
```

- AFFICHER LES VUES

  Répertoriez toutes les vues du catalogue, telles que les vues régulières, les vues multidialectales (MDV) et les vues MDV sans le dialecte Spark. La syntaxe disponible est la suivante :

  - `SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]`:

    Voici un exemple de commande permettant d'afficher les vues :

```
SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
```

Pour plus d'informations sur la création et la configuration des vues du catalogue de données, reportez-vous à la section Vues du catalogue de données Building AWS Glue dans le manuel du AWS Lake Formation développeur.

Interrogation d’un affichage du Catalogue de données

Après avoir créé une vue du catalogue de données, vous pouvez l'interroger à l'aide d'une tâche Amazon EMR Serverless Spark sur laquelle un contrôle d'accès détaillé AWS Lake Formation est activé. Le rôle d'exécution du job doit disposer de l'`SELECT`autorisation Lake Formation dans la vue du catalogue de données. Il n’est pas nécessaire d’accorder l’accès aux tables sous-jacentes référencées dans la vue.

Une fois que tout est configuré, vous pouvez interroger votre vue. Par exemple, après avoir créé une application EMR Serverless dans EMR Studio, exécutez la requête suivante pour accéder à une vue.

```
SELECT * from my_database.catalog_view LIMIT 10;
```

Une fonction utile est le`invoker_principal`. Elle renvoie l'identifiant unique du rôle d'exécution de la tâche EMRS. Cela peut être utilisé pour contrôler la sortie de la vue, en fonction du principal invoquant. Vous pouvez l'utiliser pour ajouter une condition dans votre vue qui affine les résultats de la requête, en fonction du rôle appelant. Le rôle d'exécution du job doit être autorisé à exécuter l'action `LakeFormation:GetDataLakePrincipal` IAM pour utiliser cette fonction.

```
select invoker_principal();
```

Vous pouvez ajouter cette fonction à une `WHERE` clause, par exemple, pour affiner les résultats de la requête.

Considérations et restrictions

Lorsque vous créez des vues de catalogue de données, les règles suivantes s'appliquent :

- Vous ne pouvez créer des vues de catalogue de données qu'avec Amazon EMR 7.6 et versions ultérieures.
- Le définisseur de vues du catalogue de données doit disposer de l’accès `SELECT` aux tables de base sous-jacentes accessibles par la vue. La création de la vue du catalogue de données échoue si des filtres Lake Formation sont imposés au rôle de définisseur d’une table de base spécifique.
- Les tables de base ne doivent pas être autorisées par le `IAMAllowedPrincipals` data lake dans Lake Formation. Le cas échéant, l'erreur Multi Dialect views ne peut référencer que des tables sans les autorisations IAMAllowed Principaux se produit.
- L’emplacement Amazon S3 de la table doit être enregistré en tant qu’emplacement de lac de données Lake Formation. Si la table n'est pas enregistrée, l'erreur « Multi Dialect views »

peut uniquement faire référence aux tables gérées par Lake Formation se produit. Pour plus d'informations sur la façon d'enregistrer des sites Amazon S3 à Lake Formation, reportez-vous à la section Enregistrement d'un site Amazon S3 dans le guide du AWS Lake Formation développeur.

- Vous pouvez uniquement créer des vues du catalogue de données `PROTECTED`. Les vues `UNPROTECTED` ne sont pas prises en charge.
- Vous ne pouvez pas référencer les tables d'un autre AWS compte dans une définition de vue du catalogue de données. Vous ne pouvez pas non plus référencer une table dans le même compte situé dans une région distincte.
- Pour partager des données entre un compte ou une région, la vue complète doit être partagée entre comptes et entre régions, à l'aide des liens vers les ressources de Lake Formation.
- Les fonctions définies par l'utilisateur (UDFs) ne sont pas prises en charge.
- Vous pouvez utiliser des vues basées sur des tables Iceberg. Les formats de table ouverte Apache Hudi et Delta Lake sont également pris en charge.
- Les vues du catalogue de données ne peuvent pas référencer d'autres vues.
- Un schéma de vue du catalogue de données AWS Glue est toujours stocké en minuscules. Par exemple, si vous utilisez une instruction DDL pour créer une vue Glue Data Catalog avec une colonne nomméeCastle, la colonne créée dans le Glue Data Catalog sera mise en minuscules, à. `castle` Si vous spécifiez ensuite le nom de colonne dans une requête DML sous la forme `Castle` ou`CASTLE`, EMR Spark mettra le nom en minuscules pour que vous puissiez exécuter la requête. Mais l'en-tête de colonne s'affiche en utilisant le cadre que vous avez spécifié dans la requête.

  Si vous souhaitez qu'une requête échoue dans le cas où le nom de colonne spécifié dans la requête DML ne correspond pas au nom de colonne dans le catalogue de données Glue, définissez`spark.sql.caseSensitive=true`.

Prise en charge du format de tableau ouvert

EMR Serverless prend en charge les requêtes SELECT sur Apache Hive, Apache Iceberg, Delta Lake (7.6.0+) et Apache Hudi (7.6.0+). Depuis EMR 7.12, les opérations DML et DDL qui modifient les données des tables sont prises en charge pour les tables Apache Hive, Apache Iceberg et Delta Lake à l'aide des informations d'identification vendues par Lake Formation.

Considérations et restrictions

Général

Passez en revue les limites suivantes lors de l'utilisation de Lake Formation avec EMR Serverless.

> **Note**
>
> Lorsque vous activez Lake Formation pour une tâche Spark sur EMR Serverless, la tâche lance un pilote système et un pilote utilisateur. Si vous avez spécifié une capacité préinitialisée au lancement, les pilotes sont fournis à partir de la capacité préinitialisée, et le nombre de pilotes système est égal au nombre de pilotes utilisateur que vous spécifiez. Si vous choisissez la capacité On Demand, EMR Serverless lance un pilote système en plus d'un pilote utilisateur. Pour estimer les coûts associés à votre projet EMR Serverless with Lake Formation, utilisez le. Calculateur de tarification AWS

- Amazon EMR Serverless with Lake Formation est disponible dans toutes les régions EMR sans serveur prises en charge.

- Les applications compatibles avec Lake Formation ne prennent pas en charge l'utilisation d'images EMR personnalisées sans serveur.
- Vous ne pouvez pas désactiver `DynamicResourceAllocation` les jobs de Lake Formation.
- Vous ne pouvez utiliser Lake Formation qu'avec des tâches Spark.
- EMR Serverless with Lake Formation ne prend en charge qu'une seule session Spark tout au long d'une tâche.
- EMR Serverless with Lake Formation prend uniquement en charge 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
  - Contrôle d'accès pour les colonnes imbriquées
- EMR Serverless 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, le connecteur ou le métastore Spark
  - Commande `ANALYZE TABLE`

- Si votre application EMR Serverless se trouve dans un sous-réseau privé doté de points de terminaison VPC pour Amazon S3 et que vous associez une politique de point de terminaison pour contrôler l'accès, avant que vos tâches puissent envoyer des données de journal à Amazon S3 AWS géré, incluez les autorisations détaillées dans Stockage géré dans votre politique VPC pour le point de terminaison de passerelle S3. Pour toute demande de dépannage, contactez AWS le support.
- À partir d'Amazon EMR 7.9.0, Spark FGAC prend en charge le AFile système S3 lorsqu'il est utilisé avec le schéma s3a ://.
- Amazon EMR 7.11 prend en charge la création de tables gérées à l'aide du CTAS.
- Amazon EMR 7.12 prend en charge la création de tables gérées et externes à l'aide du CTAS.

Permissions

- Pour renforcer les contrôles d'accès, les opérations EXPLAIN PLAN et DDL telles que DESCRIBE TABLE n'exposent pas d'informations restreintes.
- Lorsque vous enregistrez l'emplacement d'une table auprès de Lake Formation, l'accès aux données utilise les informations d'identification stockées par Lake Formation au lieu des autorisations IAM du rôle d'exécution des tâches EMR Serverless. Les tâches échoueront si le rôle enregistré pour l'emplacement de la table est mal configuré, même si le rôle d'exécution dispose des autorisations IAM S3 pour cet emplacement.
- À partir d'Amazon EMR 7.12, vous pouvez écrire dans des tables Hive et Iceberg existantes en utilisant DataFrameWriter (V2) avec les informations d'identification de Lake Formation en mode ajout. Pour les opérations de remplacement ou lors de la création de nouvelles tables, EMR utilise les informations d'identification du rôle d'exécution pour modifier les données des tables.
- Les limitations suivantes s'appliquent lors de l'utilisation de vues ou de tables mises en cache comme données source (ces limitations ne s'appliquent pas aux vues du catalogue de données AWS Glue) :
  - Pour les opérations MERGE, DELETE et UPDATE
    - Supporté : utilisation de vues et de tables mises en cache comme tables sources.
    - Non pris en charge : utilisation de vues et de tables mises en cache dans les clauses d'affectation et de condition.
  - Pour les opérations CREATE OR REPLACE et REPLACE TABLE AS SELECT :
    - Non pris en charge : utilisation de vues et de tables mises en cache comme tables sources.

- Les tables Delta Lake contenant des données source prennent UDFs en charge les opérations MERGE, DELETE et UPDATE uniquement lorsque le vecteur de suppression est activé.

Journaux et débogage

- EMR Serverless restreint l'accès aux journaux Spark du pilote système sur les applications compatibles avec Lake Formation. Étant donné que le pilote système fonctionne avec des autorisations élevées, 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 Serverless désactive l'accès aux journaux des pilotes du système.
- Les journaux des profils système sont toujours conservés dans le stockage géré. Il s'agit d'un paramètre obligatoire qui ne peut pas être désactivé. Ces journaux sont stockés de manière sécurisée et chiffrés à l'aide d'une clé KMS gérée par le client ou d'une clé KMS AWS gérée.

Iceberg

Prenez en compte les points suivants 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`,, `files`,`manifests`, et`refs`. Amazon EMR masque les colonnes susceptibles de contenir des données sensibles, telles que `partitions`,`path`, et. `summaries` Cette restriction ne s’applique pas aux tables Iceberg qui ne sont pas enregistrées dans Lake Formation.
- Les tables qui ne sont pas enregistrées 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 conseillons d'utiliser Iceberg DataFrameWriter V2 au lieu de V1.

Résolution des problèmes

Consultez les sections suivantes pour obtenir des solutions de dépannage.

Logging

EMR Serverless utilise les profils de ressources Spark pour diviser l'exécution des tâches. EMR Serverless 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 tâches exécutées en tant que profil utilisateur.

Pour plus d'informations sur le débogage des tâches compatibles avec Lake Formation, reportez-vous à la section Débogage des tâches.

Interface utilisateur Live et serveur d’historique Spark

L’interface utilisateur Live et le serveur d’historique Spark contiennent tous les événements Spark générés à partir du profil utilisateur et les événements expurgés générés à partir du pilote système.

Vous pouvez voir toutes les tâches des pilotes utilisateur et système dans l'onglet Exécuteurs. Toutefois, les liens du journal ne sont disponibles que pour le profil utilisateur. Certaines informations sont également expurgées de l’’interface utilisateur Live, comme le nombre d’enregistrements de sortie.

Échec de la tâche en raison d’autorisations insuffisantes pour Lake Formation

Assurez-vous que votre rôle d’exécution des tâches dispose des autorisations nécessaires pour exécuter SELECT et DESCRIBE sur la table à laquelle vous accédez.

Échec de l’exécution de la tâche avec RDD

EMR Serverless 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 Serverless 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, reportez-vous aux sections 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.

API native de contrôle d'accès affinée de Spark (alllowlisted) PySpark

Pour garantir la sécurité et le contrôle de l'accès aux données, le contrôle d'accès fin (FGAC) de Spark restreint certaines fonctions. PySpark Ces restrictions sont appliquées par le biais de :

- Blocage explicite qui empêche l'exécution de fonctions
- Incompatibilités d'architecture qui rendent les fonctions non fonctionnelles
- Fonctions susceptibles de générer des erreurs, de renvoyer des messages d'accès refusé ou de ne rien faire lorsqu'elles sont appelées

Les PySpark fonctionnalités suivantes ne sont pas prises en charge dans Spark FGAC :

- Opérations RDD (bloquées par Spark RDDUnsupported Exception)
- Spark Connect (non pris en charge)
- Spark Streaming (non pris en charge)

Bien que nous ayons testé les fonctions répertoriées dans un environnement Native Spark FGAC et confirmé qu'elles fonctionnent comme prévu, nos tests ne portent généralement que sur l'utilisation de base de chaque API. Les fonctions comportant plusieurs types d'entrées ou des chemins logiques complexes peuvent présenter des scénarios non testés.

Pour les fonctions qui ne sont pas répertoriées ici et qui ne font pas clairement partie des catégories non prises en charge ci-dessus, nous vous recommandons de :

- Les tester d'abord dans un environnement gamma ou dans un déploiement à petite échelle
- Vérifier leur comportement avant de les utiliser en production

> **Note**
>
> Si vous voyez une méthode de classe répertoriée mais pas sa classe de base, la méthode devrait toujours fonctionner. Cela signifie simplement que nous n'avons pas vérifié explicitement le constructeur de la classe de base.

L' PySpark API est organisée en modules. Le support général pour les méthodes de chaque module est détaillé dans le tableau ci-dessous.

Nom du module	Statut	Remarques
pyspark_core	Pris en charge	Ce module contient les principales classes RDD, et ces fonctions ne sont généralement pas prises en charge.
pyspark_sql	Pris en charge	
pyspark_testing	Pris en charge	
pyspark_resource	Pris en charge	
pyspark_streaming	Bloqué	L'utilisation du streaming est bloquée dans Spark FGAC.
pyspark_mllib	Experimental	Ce module contient des opérations ML basées sur RDD, et ces fonctions ne sont généralement pas prises en charge. Ce module n'a pas fait l'objet de tests approfondis.
pyspark_ml	Experimental	Ce module contient des opérations ML DataFrame basées sur le langage machine, et ces fonctions sont pour la plupart prises en charge. Ce module n'a pas fait l'objet de tests approfond is.

Nom du module	Statut	Remarques
pyspark_pandas	Pris en charge	
pyspark_pandas_slow	Pris en charge	
pyspark_connect	Bloqué	L'utilisation de Spark Connect est bloquée dans Spark FGAC.
pyspark_pandas_connect	Bloqué	L'utilisation de Spark Connect est bloquée dans Spark FGAC.
pyspark_pandas_slow_connect	Bloqué	L'utilisation de Spark Connect est bloquée dans Spark FGAC.
pyspark_errors	Experimental	Ce module n'a pas fait l'objet de tests approfondis. Les classes d'erreur personnalisées ne peuvent pas être utilisées .

Liste des API autorisées

Pour une liste téléchargeable et plus facile à rechercher, un fichier contenant les modules et les classes est disponible dans les fonctions Python autorisées dans Native FGAC.

Chiffrement inter-travailleurs

Avec les versions 6.15.0 et supérieures d'Amazon EMR, activez les communications cryptées par protocole TLS mutuel entre les collaborateurs lors de vos exécutions de tâches Spark. Lorsqu'il est activé, EMR Serverless génère et distribue automatiquement un certificat unique pour chaque travailleur approvisionné dans le cadre de vos cycles de travail. Lorsque ces travailleurs communiquent pour échanger des messages de contrôle ou transférer des données de shuffle, ils établissent une connexion TLS mutuelle et utilisent les certificats configurés pour vérifier l'identité des

autres. Si un travailleur n'est pas en mesure de vérifier un autre certificat, le handshake TLS échoue et EMR Serverless interrompt la connexion entre eux.

Si vous utilisez Lake Formation avec EMR Serverless, le chiffrement TLS mutuel est activé par défaut.

Activation du chiffrement TLS mutuel sur EMR Serverless

Pour activer le chiffrement TLS mutuel sur votre application Spark, définissez ce paramètre sur true lors `spark.ssl.internode.enabled` de la création de l'application EMR Serverless. Si vous utilisez la AWS console pour créer une application EMR sans serveur, choisissez Utiliser les paramètres personnalisés, puis développez la configuration de l'application et entrez votre. `runtimeConfiguration`

```
aws emr-serverless create-application \
--release-label emr-6.15.0 \
--runtime-configuration '{
  "classification": "spark-defaults",
  "properties": {"spark.ssl.internode.enabled": "true"}
}' \
--type "SPARK"
```

Si vous souhaitez activer le chiffrement TLS mutuel pour les exécutions de tâches Spark individuelles, définissez ce paramètre sur `spark.ssl.internode.enabled` true lors de l'utilisation`spark-submit`.

```
--conf spark.ssl.internode.enabled=true
```

Chiffrement de disque avec KMS CMK

EMR Serverless chiffre tous les disques connectés aux ordinateurs par défaut à l'aide de clés de chiffrement appartenant au service. Vous pouvez éventuellement choisir de chiffrer ces disques à l'aide de vos propres clés gérées par AWS KMS le client (CMKs). Cela vous permet de mieux contrôler vos clés de chiffrement, notamment d'établir et de maintenir des politiques relatives aux clés, ainsi que d'auditer l'utilisation des clés.

Vous pouvez configurer le chiffrement du disque soit lors de la création d'une application, soit lors de la soumission de tâches individuelles. Lorsqu'elles sont activées au niveau de l'application, toutes

les tâches de cette application héritent des paramètres de chiffrement. Vous pouvez également remplacer la valeur par défaut de l'application en spécifiant une configuration de chiffrement de disque lors de la soumission d'une tâche.

> **Note**
>
> Le chiffrement de disque EMR sans serveur ne prend en charge que les clés KMS symétriques. Les clés KMS asymétriques ne sont pas prises en charge. Vous devez utiliser une clé KMS de chiffrement symétrique créée dans AWS KMS. Pour plus d'informations AWS KMS, voir Qu'est-ce que c'est AWS KMS ?

Utilisation du contexte de chiffrement

EMR Serverless utilise éventuellement un contexte de chiffrement pour fournir des données authentifiées supplémentaires pour les opérations de chiffrement. Le contexte de chiffrement est un ensemble de paires clé-valeur qui peuvent contenir des données authentifiées supplémentaires non secrètes. Le contexte de chiffrement étant lié cryptographiquement aux données chiffrées, le même contexte de chiffrement est requis pour déchiffrer les données.

Dans EMR Serverless, vous pouvez spécifier le contexte de chiffrement personnalisé lors de la configuration du chiffrement du disque. Ce contexte de chiffrement est inclus dans AWS CloudTrail les journaux pour vous aider à identifier et à comprendre vos opérations KMS.

> **Note**
>
> Ne stockez pas d'informations sensibles dans un contexte de chiffrement tel qu'elles apparaissent en texte clair dans les AWS CloudTrail journaux.

Configuration du chiffrement des disques à l'aide de clés gérées par le client

CreateApplication

Pour chiffrer des disques avec votre propre clé KMS, incluez le `diskEncryptionConfiguration` paramètre lors de la création d'une application EMR Serverless.

```
aws emr-serverless create-application \
```

```
  --type TYPE \
  --name APPLICATION_ID \
  --release-label RELEASE_LABEL \
  --region AWS_REGION \
  --disk-encryption-configuration '{
        "encryptionKeyArn": "key-arn",
        "encryptionContext": {
            "key": "value"
        }
  }'
```

UpdateApplication

Pour mettre à jour le contexte de and/or chiffrement ARN de la clé KMS, spécifiez le `diskEncryptionConfiguration` paramètre avec les nouvelles valeurs lors de la mise à jour d'une application.

```
aws emr-serverless update-application \
  --name APPLICATION_ID \
  --region AWS_REGION \
  --disk-encryption-configuration '{
        "encryptionKeyArn": "key-arn",
        "encryptionContext": {
            "key": "value"
        }
  }'
```

> **Note**
>
> Pour annuler le chiffrement de disque configuré sur une application, transmettez une application vide `diskEncryptionConfiguration` lors de la mise à jour.

StartJobRun

Pour chiffrer des disques avec votre propre clé KMS, utilisez la `diskEncryptionConfiguration` configuration lorsque vous soumettez une exécution de tâche.

```
--configuration-overrides '{
        "diskEncryptionConfiguration": {
```

```
            "encryptionKeyArn": "key-arn",
            "encryptionContext": {
                "key": "value"
            }
        }
    }'
```

Points de terminaison Public Livy

Pour chiffrer des disques avec votre propre clé KMS lors de la création de sessions Spark via des points de terminaison publics Livy, spécifiez la configuration de chiffrement dans l'objet de la session. `conf`

```
data = {
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "role_arn",
        "spark.emr-serverless.disk.encryptionKeyArn": "key-arn",
        "spark.emr-serverless.disk.encryptionContext": "key1:value1,key2:value2"  #
 Optional
    }
}

# Send request to create a session with the Livy API endpoint
request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
 headers=headers)
```

Autorisations requises pour le chiffrement du disque

Autorisations relatives aux clés de chiffrement pour EMR Serverless

Lorsque vous chiffrez des disques avec votre propre clé de chiffrement, vous devez configurer les autorisations de clé KMS suivantes pour le `emr-serverless.amazonaws.com` principal :

- `kms:GenerateDataKey`: pour générer des clés de données afin de chiffrer les volumes de disque
- `kms:Decrypt`: pour déchiffrer les clés de données lors de l'accès au contenu chiffré du disque

```
{
```

```
    "Effect": "Allow",
    "Principal":{
        "Service": "emr-serverless.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:applicationId": "application-id",
            "aws:SourceAccount": "aws-account-id"
        }
    }
}
```

Pour des raisons de sécurité, nous vous recommandons d'ajouter une clé de `aws:SourceArn` condition à la politique de clé KMS. La clé de condition globale IAM `aws:SourceArn` permet de garantir qu'EMR Serverless utilise la clé KMS uniquement pour l'ARN d'une application. En outre, l'inclusion de la clé de `aws:SourceAccount` condition fournit un niveau de sécurité supplémentaire en limitant l'utilisation de votre clé KMS aux demandes provenant de l'identifiant de AWS compte spécifié dans la condition.

Le rôle d'exécution des tâches doit disposer des autorisations suivantes dans sa politique IAM :

```
{
    "Sid": "Enable GDK and Decrypt",
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "key-arn"
    }
}
```

Autorisations utilisateur requises

L'utilisateur qui soumet la tâche doit être autorisé à utiliser la clé. Vous pouvez spécifier les autorisations dans la politique clé KMS ou dans la stratégie IAM pour l'utilisateur, le groupe ou le rôle. Si l'utilisateur qui soumet la tâche ne possède pas les autorisations nécessaires à la clé KMS, EMR Serverless rejette la soumission de l'exécution de la tâche.

Exemple de politique de clé

La politique clé suivante fournit les autorisations nécessaires pour `kms:DescribeKey` `kms:GenerateDataKey` et `kms:Decrypt` :

- `kms:DescribeKey`: pour vérifier que la clé KMS gérée par le client est activée et SYMÉTRIQUE avant de l'utiliser.

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal":{
        "AWS": "arn:aws:iam::111122223333:user/user-name"
    },
    "Action": [
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Enable GDK and Decrypt",
    "Effect": "Allow",
    "Principal":{
        "AWS": "arn:aws:iam::111122223333:user/user-name"
    },
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": "emr-serverless.region.amazonaws.com",
            "kms:EncryptionContext:key": "value"
        }
```

```
    }
}
```

Pour des raisons de sécurité, nous vous recommandons d'ajouter une clé de `kms:viaService` condition à la politique de clé KMS. Il limite l'utilisation de la clé KMS aux demandes de validation provenant uniquement d'emr-serverless.

Exemple de politique IAM

La politique IAM suivante fournit les autorisations pour`kms:DescribeKey`, `kms:GenerateDataKey` et`kms:Decrypt`.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "kms:DescribeKey",
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "key-arn"
    }
}
```

Surveillance de l'utilisation des clés

Vous pouvez surveiller l'utilisation des clés gérées par vos clients dans EMR Serverless via. AWS CloudTrail AWS CloudTrail capture tous les appels d'API AWS KMS sous forme d'événements, y compris les appels provenant de la console EMR Serverless, de l'API EMR Serverless, de la CLI ou du SDK. AWS AWS

Les informations capturées incluent le contexte de chiffrement que vous avez spécifié, qui peut vous aider à identifier et à auditer les ressources EMR Serverless spécifiques qui ont utilisé votre clé KMS. Par exemple, vous pouvez voir des événements similaires aux suivants dans AWS CloudTrail. Pour plus d'informations sur l'utilisation AWS CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

GenerateDataKey

Exemple d'événement pour les GenerateDataKey opérations lorsque EMR Serverless crée des volumes de disque chiffrés

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "principalId": "user",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-07-28T21:43:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ipAddress",
    "userAgent": "userAgent",
    "requestParameters": {
        "encryptionContext": {
            "applicationId": "test"
        },
        "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
    },
    "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
    "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "accountId",
    "eventCategory": "Management"
}
```

Decrypt

Exemple d'événement pour les opérations de déchiffrement lorsque EMR Serverless accède à des données chiffrées.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "principalId": "user",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-07-28T21:43:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ipAddress",
    "userAgent": "userAgent",
    "requestParameters": {
        "encryptionContext": {
            "applicationId": "test"
        },
        "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
    },
    "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
    "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "accountId",
```

```
    "eventCategory": "Management"
}
```

En savoir plus

Les ressources suivantes fournissent plus d'informations sur le chiffrement des données au repos.

- Pour plus d'informations sur AWS KMS les concepts de base, consultez le guide du AWS KMS développeur.
- Pour plus d'informations sur les meilleures pratiques de sécurité pour AWS KMS, consultez le guide du AWS KMS développeur.

Secrets Manager pour la protection des données avec EMR Serverless

AWS Secrets Manager est un service de stockage secret destiné à protéger les informations d'identification des bases de données, les clés d'API et d'autres informations secrètes. Ensuite, dans votre code, remplacez les informations d'identification codées en dur par un appel d'API à Secrets Manager. Cela permet de s'assurer que le secret ne peut pas être compromis par quelqu'un qui examine votre code, car le secret n'existe pas. Pour un aperçu, reportez-vous au guide de l'AWS Secrets Manager utilisateur.

Secrets Manager chiffre les secrets à l'aide de AWS Key Management Service clés. Pour plus d'informations, reportez-vous à la section Chiffrement et déchiffrement secrets du guide de l'AWS Secrets Manager utilisateur.

Vous pouvez configurer Secrets Manager pour qu'il fasse automatiquement pivoter les secrets selon un calendrier que vous spécifiez. Cela vous permet de remplacer les secrets à long terme par ceux à court terme, ce qui réduit considérablement le risque de mise en péril. Pour plus d'informations, reportez-vous à la section Rotation AWS Secrets Manager des secrets dans le guide de AWS Secrets Manager l'utilisateur.

Amazon EMR Serverless s'intègre AWS Secrets Manager afin que vous puissiez stocker vos données dans Secrets Manager et utiliser l'identifiant secret dans vos configurations.

Comment EMR Serverless utilise les secrets

Lorsque vous stockez vos données dans Secrets Manager et que vous utilisez l'identifiant secret dans vos configurations pour EMR Serverless, vous ne transmettez pas de données de configuration sensibles à EMR Serverless en texte brut et vous ne les exposez pas à des données externes. APIs Si vous indiquez qu'une paire clé-valeur contient un identifiant secret pour un secret que vous avez stocké dans Secrets Manager, EMR Serverless récupère le secret lorsqu'il envoie des données de configuration aux opérateurs chargés d'exécuter des tâches.

Pour indiquer qu'une paire clé-valeur d'une configuration contient une référence à un secret stocké dans Secrets Manager, ajoutez l'`EMR.secret@`annotation à la valeur de configuration. Pour toute propriété de configuration comportant une annotation d'identifiant secret, EMR Serverless appelle Secrets Manager et résout le secret au moment de l'exécution de la tâche.

Comment créer un secret

Pour créer un secret, suivez les étapes décrites dans Créer un AWS Secrets Manager secret dans le Guide de AWS Secrets Manager l'utilisateur. À l'étape 3, choisissez le champ de texte en clair pour saisir votre valeur sensible.

Fournir un secret dans une classification de configuration

Les exemples suivants montrent comment fournir un secret dans une classification de configuration à`StartJobRun`. Si vous souhaitez configurer les classifications pour Secrets Manager au niveau de l'application, reportez-vous àConfiguration d'application par défaut pour EMR Serverless.

Dans les exemples, remplacez *`SecretName`* par le nom du secret à récupérer. Pour plus d'informations, consultez Comment créer un secret.

Dans cette section

- Spécifier des références secrètes - Spark
- Spécifier des références secrètes - Hive

Spécifier des références secrètes - Spark

Example— Spécifiez des références secrètes dans la configuration du métastore Hive externe pour Spark

```
aws emr-serverless start-job-run \
```

```
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-
connector-java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=connection-user-
name
            --conf
 spark.hadoop.javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

Example— Spécifiez des références secrètes pour la configuration de la métastore Hive externe dans la classification spark-defaults

```
{
        "classification": "spark-defaults",
        "properties": {

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver"
            "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name"
            "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-name"
            "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretName",
        }
```

```
    }
```

Spécifier des références secrètes - Hive

Example— Spécifiez des références secrètes dans la configuration du métastore Hive externe pour Hive

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
    --job-driver '{
        "hive": {
        "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
 javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
 hive.metastore.client.factory.class=org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreCli
                    --hiveconf
 javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
                    --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket"
        }
    }
}'
```

Example— Spécifiez des références secrètes pour la configuration de la métastore Hive externe dans la classification hive-site

```
{
    "classification": "hive-site",
    "properties": {
```

```
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "EMR.secret@SecretName"
    }
}
```

Accordez l'accès à EMR Serverless pour récupérer le secret

Pour permettre à EMR Serverless de récupérer la valeur secrète depuis Secrets Manager, ajoutez la déclaration de politique suivante à votre secret lorsque vous le créez. Vous devez créer votre secret à l'aide de la clé KMS gérée par le client pour qu'EMR Serverless puisse lire la valeur du secret. Pour plus d'informations, reportez-vous à la section Autorisations relatives à la clé KMS dans le guide de AWS Secrets Manager l'utilisateur.

Dans la politique suivante, remplacez-le *applicationId* par l'ID de votre application.

Politique en matière de ressources pour le secret

Vous devez inclure les autorisations suivantes dans la politique de ressources pour le secret afin de permettre AWS Secrets Manager à EMR Serverless de récupérer les valeurs secrètes. Pour garantir que seule une application spécifique peut récupérer ce secret, vous pouvez éventuellement spécifier l'ID de l'application EMR Serverless comme condition dans la politique.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
```

```
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
*"
        }
      },
      "Sid": "AllowSECRETSMANAGERGetsecretvalue"
    }
  ]
}
```

Créez votre secret avec la politique suivante pour la clé gérée par le client AWS Key Management Service (AWS KMS) :

Politique relative aux clés gérées par le client AWS KMS

```
{
    "Sid": "Allow EMR Serverless to use the key for decrypting secrets",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "emr-serverless.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": "secretsmanager.Région AWS.amazonaws.com"
        }
    }
}
```

Transformer le secret

La rotation se produit lorsque vous mettez régulièrement à jour un secret. Vous pouvez le configurer AWS Secrets Manager pour faire automatiquement pivoter le secret selon un calendrier que vous spécifiez. De cette façon, vous pouvez remplacer les secrets à long terme par des secrets à court

terme. Cela permet de réduire le risque de compromission. EMR Serverless récupère la valeur secrète d'une configuration annotée lorsque la tâche passe à l'état en cours d'exécution. Si vous ou un processus mettez à jour la valeur secrète dans Secrets Manager, vous devez soumettre une nouvelle tâche afin que la tâche puisse récupérer la valeur mise à jour.

> **Note**
>
> Les tâches déjà en cours d'exécution ne peuvent pas récupérer une valeur secrète mise à jour. Cela peut entraîner un échec professionnel.

Utilisation des autorisations d'accès Amazon S3 avec EMR sans serveur

Présentation des subventions d'accès S3 pour EMR Serverless

Avec les versions 6.15.0 et supérieures d'Amazon EMR, Amazon S3 Access Grants fournit une solution de contrôle d'accès évolutive pour augmenter l'accès à vos données Amazon S3 depuis EMR Serverless. Si vous disposez d'une configuration d'autorisations complexe ou importante pour vos données S3, utilisez Access Grants pour adapter les autorisations de données S3 aux utilisateurs, aux rôles et aux applications.

Utilisez les subventions d'accès S3 pour augmenter l'accès aux données Amazon S3 au-delà des autorisations accordées par le rôle d'exécution ou des rôles IAM associés aux identités ayant accès à votre application EMR Serverless.

Pour plus d'informations, reportez-vous aux sections Gestion de l'accès avec S3 Access Grants pour Amazon EMR dans le Guide de gestion Amazon EMR et Gestion de l'accès avec S3 Access Grants dans le Guide de l'utilisateur d'Amazon Simple Storage Service.

Cette section explique comment lancer une application EMR sans serveur qui utilise les autorisations d'accès S3 pour fournir un accès aux données dans Amazon S3. Pour savoir comment utiliser 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 Amazon EMR sur EKS

Lancez une application EMR sans serveur avec S3 Access Grants pour la gestion des données

Vous pouvez activer S3 Access Grants sur EMR Serverless et lancer une application 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 tâche pour votre application EMR Serverless. Incluez les autorisations IAM requises pour exécuter des tâches Spark et utiliser S3 Access Grants, `s3:GetDataAccess` et `s3:GetAccessGrantsInstanceForPrefix` :

```
{
    "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 pour l'exécution des tâches dotés d'autorisations supplémentaires pour accéder directement à S3, les utilisateurs peuvent accéder aux données autorisées par le rôle même s'ils n'ont pas l'autorisation de S3 Access Grants.

2. Lancez votre application EMR Serverless avec une étiquette de version Amazon EMR 6.15.0 ou supérieure et la `spark-defaults` classification, comme le montre l'exemple suivant. Remplacez les valeurs dans *`red text`* par les valeurs appropriées pour votre scénario d'utilisation.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
        "sparkSubmit": {
```

```
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket1/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
 --conf spark.executor.instances=1"
          }
    }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
          "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }]
}'
```

Considérations relatives aux autorisations d'accès S3 avec EMR Serverless

Pour obtenir des informations importantes sur l'assistance, la compatibilité et le comportement lorsque vous utilisez Amazon S3 Access Grants avec EMR Serverless, reportez-vous aux considérations relatives aux subventions d'accès S3 avec Amazon EMR dans le guide de gestion Amazon EMR.

Journalisation des appels d'API sans serveur Amazon EMR à l'aide de AWS CloudTrail

Amazon EMR Serverless est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un service AWS dans EMR Serverless. CloudTrail capture tous les appels d'API pour EMR Serverless sous forme d'événements. Les appels capturés incluent des appels provenant de la console EMR Serverless et des appels de code vers les opérations de l'API EMR Serverless. Si vous créez un suivi, activez la diffusion continue des CloudTrail événements vers un compartiment Amazon S3, y compris les événements pour EMR Serverless. Si vous ne configurez pas de suivi, vous pouvez toujours accéder aux événements les plus récents de la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à EMR Serverless,

l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires.

Pour en savoir plus, consultez le guide de AWS CloudTrail l'utilisateur.

Informations EMR Serverless dans CloudTrail

CloudTrail est activé sur votre compte Compte AWS lorsque vous créez le compte. Lorsqu'une activité se produit dans EMR Serverless, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez accéder aux événements récents, les rechercher et les télécharger dans votre Compte AWS. Pour plus d'informations, reportez-vous à la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement continu des événements survenus dans votre environnement Compte AWS, y compris les événements relatifs à EMR Serverless, créez une trace. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s'applique à toutes les régions Régions AWS. 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, configurez d'autres AWS services pour analyser et agir de manière plus approfondie sur les données d'événements collectées dans CloudTrail les journaux. Pour plus d'informations, reportez-vous aux rubriques suivantes :

- Présentation de la création d'un journal d'activité
- CloudTrail services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions EMR Serverless sont enregistrées CloudTrail et documentées dans le manuel EMR Serverless API Reference. Par exemple, les appels au `CreateApplication` `StartJobRun` et les `CancelJobRun` 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, reportez-vous à l'élément CloudTrail UserIdentity.

Comprendre les entrées du fichier journal EMR sans serveur

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'`CreateApplication`action.

```
{
    "eventVersion": "1.08",
    "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": {
                "creationDate": "2022-06-01T23:46:52Z",
                "mfaAuthenticated": "false"
            }
        }
    },
```

```
    "eventTime": "2022-06-01T23:49:28Z",
    "eventSource": "emr-serverless.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.26.10",
    "requestParameters": {
        "name": "my-serverless-application",
        "releaseLabel": "emr-6.6",
        "type": "SPARK",
        "clientToken": "0a1b234c-de56-7890-1234-567890123456"
    },
    "responseElements": {
        "name": "my-serverless-application",
        "applicationId": "1234567890abcdef0",
        "arn": "arn:aws:emr-serverless:us-west-2:555555555555:/
applications/1234567890abcdef0"
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management"
}
```


Validation de conformité pour Amazon EMR Serverless

La sécurité et la conformité d'EMR Serverless sont évaluées par des auditeurs tiers dans le cadre de plusieurs programmes de AWS conformité, notamment les suivants :

- System and Organization Controls (SOC)
- Norme de sécurité des données de l’industrie des cartes de paiement (PCI DSS)
- Programme fédéral de gestion des risques et des autorisations (FedRAMP) modéré
- Health Insurance Portability and Accountability Act (HIPAA)

AWS fournit une liste fréquemment mise à jour des AWS services concernés par des programmes de conformité spécifiques sur la page AWS Services in Scope by Compliance Program.

Les rapports d'audit tiers peuvent être téléchargés à l'aide de AWS Artifact. Pour plus d'informations, reportez-vous à la section Téléchargement de rapports dans AWS Artifact.

Pour plus d'informations sur les programmes de AWS conformité, reportez-vous à la section Programmes de AWS conformité.

Lorsque vous utilisez EMR Serverless, votre responsabilité en matière de conformité dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. Si votre utilisation d'EMR Serverless est soumise à la conformité à des normes telles que HIPAA, PCI ou FedRAMP Moderate, fournit des ressources pour vous aider à : AWS

- Guides de démarrage rapide sur la sécurité et la conformité qui abordent les considérations architecturales et les étapes à suivre pour déployer des environnements de base axés sur la sécurité et la conformité sur. AWS
- AWS Les guides de conformité destinés aux clients peuvent vous aider à comprendre le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques pour sécuriser les Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (y compris l'Institut national de normalisation et de technologie (NIST), le Conseil de normes de sécurité PCI (Payment Card Industry) et l'Organisation internationale de normalisation (ISO)).
- Le service AWS Config peut permettre d'évaluer comment les configurations de vos ressources se conforment aux pratiques internes, aux normes et aux directives industrielles.
- AWS Les ressources de conformité sont un ensemble de classeurs et de guides qui peuvent s'appliquer à votre secteur d'activité et à votre région.
- AWS Security Hub vous fournit une vue complète de l'état de votre sécurité interne AWS et vous aide à vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité.
- AWS Audit Manager— cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Résilience dans Amazon EMR Serverless

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, concevez et gérez des applications et des bases de données qui basculent automatiquement entre les zones 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, reportez-vous à la section Infrastructure AWS mondiale.

Outre l'infrastructure AWS mondiale, Amazon EMR Serverless propose une intégration avec Amazon S3 via EMRFS pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Sécurité de l'infrastructure dans Amazon EMR Serverless

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 configuration et de vulnérabilité dans Amazon EMR Serverless

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 conformité pour Amazon EMR Serverless
- Modèle de responsabilité partagée

- Amazon Web Services : Présentation des procédures de sécurité

Points de terminaison et quotas pour EMR Serverless

Points de terminaison de service

Pour vous connecter par programmation à un Service AWS, vous utilisez un point de terminaison. Un point de terminaison est l'URL du point d'entrée d'un service AWS Web. Outre les points de AWS terminaison standard, certains Services AWS proposent des points de terminaison FIPS dans certaines régions. Le tableau suivant répertorie les points de terminaison de service pour EMR Serverless. Pour plus d'informations, reportez-vous à la section Service AWS Points de terminaison.

Points de terminaison de service EMR sans serveur

Nom de la région	Région	Point de terminaison	Protocole
USA Est (Ohio)	`us-east-2` (limité aux zones de disponibilité suivantes : `use2-az1use2-az2`, et`use2-az3`)	`emr-serverless.us-east-2.amazonaws.com`	HTTPS

USA Est (Virginie du Nord)	`us-east-1` (limité aux zones de disponibilité suivantes : `use1-az1use1-az2`,`use1-az4`,`use1-az5`, et`use1-az6`)	`emr-serverless.us-east-1.amazonaws.com`
		
		`emr-serverless-fips.us-east-1.amazonaws.com`
			HTTPS

USA Ouest (Californie du Nord)	`us-west-1`	`emr-serverless.us-west-1.amazonaws.com`	HTTPS
USA Ouest (Oregon)	`us-west-2`	`emr-serverless.us-`	HTTPS

Nom de la région	Région	Point de terminaison	Protocole

		`west-2.amazonaws.com`
		
		`emr-serverless-fips.us-west-2.amazonaws.com`	

Afrique (Le Cap)	`af-south-1`	`emr-serverless.af-south-1.amazonaws.com`	HTTPS
Asie-Pacifique (Hong Kong)	`ap-east-1`	`emr-serverless.ap-east-1.amazonaws.com`	HTTPS
Asie-Pacifique (Jakarta)	`ap-southeast-3`	`emr-serverless.ap-southeast-3.amazonaws.com`	HTTPS
Asie-Pacifique (Melbourne)	`ap-southeast-4`	`emr-serverless.ap-southeast-4.amazonaws.com`	HTTPS
Asie-Pacifique (Malaisie)	`ap-southeast-5`	`emr-serverless.ap-southeast-5.amazonaws.com`	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Asie-Pacifique (Mumbai)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
Asie-Pacifique (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS
Asie-Pacifique (Séoul)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
Asie-Pacifique (Singapour)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
Asie-Pacifique (Sydney)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
Asie-Pacifique (Tokyo)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Canada (Centre)	`ca-centra l-1` (limité aux zones de disponibilité suivantes : `cac1-az1` et`cac1-az2`)	`emr-serve rless.ca- central-1 .amazonaws.com`	HTTPS
Canada-Ouest (Calgary)	`ca-west-1`	`emr-serve rless.ca- west-1.am azonaws.com`	HTTPS
Europe (Francfort)	`eu-central-1`	`emr-serve rless.eu- central-1 .amazonaws.com`	HTTPS
Europe (Zurich)	`eu-central-2`	`emr-serve rless.eu- central-2 .amazonaws.com`	HTTPS
Europe (Irlande)	`eu-west-1`	`emr-serve rless.eu- west-1.am azonaws.com`	HTTPS
Europe (Londres)	`eu-west-2`	`emr-serve rless.eu- west-2.am azonaws.com`	HTTPS
Europe (Milan)	`eu-south-1`	`emr-serve rless.eu- south-1.a mazonaws.com`	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Europe (Paris)	eu-west-3	emr-serverless.eu-west-3.amazonaws.com	HTTPS
Europe (Espagne)	eu-south-2	emr-serverless.eu-south-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
Israël (Tel Aviv)	il-central-1	emr-serverless.il-central-1.amazonaws.com	HTTPS
Middle East (Bahrain)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS
Moyen-Orient (EAU)	me-central-1	emr-serverless.me-central-1.amazonaws.com	HTTPS
Amérique du Sud (São Paulo)	sa-east-1	emr-serverless.sa-east-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Chine (Pékin)	`cn-north-1` (limité aux zones de disponibilité suivantes :`cnn1-az1`,`cnn1-az2`)	`emr-serverless.cn-north-1.amazonaws.com.cn`	HTTPS
AWS GovCloud (USA Est)	`us-gov-east-1`	`emr-serverless.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (US-Ouest)	`us-gov-west-1`	`emr-serverless.us-gov-west-1.amazonaws.com`	HTTPS

Quotas de service

Les quotas de service, également appelés limites, sont le nombre maximum de ressources de service ou d'opérations que vous Compte AWS pouvez utiliser. EMR Serverless collecte les métriques d'utilisation des quotas de service toutes les minutes et les publie dans l'espace de noms. `AWS/Usage`

> **Note**
>
> AWS Les nouveaux comptes ont des quotas initiaux plus bas qui peuvent augmenter au fil du temps. Amazon EMR Serverless surveille l'utilisation des comptes au sein de chaque compte Région AWS, puis augmente automatiquement les quotas en fonction de votre utilisation.

Le tableau suivant répertorie les quotas de service pour EMR Serverless. Pour plus d'informations, reportez-vous à la section Service AWS Quotas.

Nom	Limite par défaut	Ajustable?	Description
Nombre maximal de v simultanés CPUs par compte	16	Oui	Le nombre maximum de v CPUs qui peuvent être exécutés simultanément pour le compte en cours Région AWS.
Nombre maximum de tâches en file d'attente par compte	2000	Oui	Le nombre maximum de tâches en file d'attente pour le compte en cours. Région AWS

Limites d'API

Ce qui suit décrit les limites d'API par région pour votre Compte AWS.

Ressource	Quota par défaut
ListApplications	10 transactions par seconde. Une rafale de 50 transactions par seconde.
CreateApplication	1 transaction par seconde. Une rafale de 25 transactions par seconde.
DeleteApplication	1 transaction par seconde. Une rafale de 25 transactions par seconde.
GetApplication	10 transactions par seconde. Une rafale de 50 transactions par seconde.
UpdateApplication	1 transaction par seconde. Une rafale de 25 transactions par seconde.

Ressource	Quota par défaut
ListJobRuns	1 transaction par seconde. Une rafale de 25 transactions par seconde.
StartJobRun	1 transaction par seconde. Une rafale de 25 transactions par seconde.
GetDashboardForJobRun	1 transaction par seconde. Une rafale de 2 transactions par seconde.
CancelJobRun	1 transaction par seconde. Une rafale de 25 transactions par seconde.
GetJobRun	10 transactions par seconde. Une rafale de 50 transactions par seconde.
StartApplication	1 transaction par seconde. Une rafale de 25 transactions par seconde.
StopApplication	1 transaction par seconde. Une rafale de 25 transactions par seconde.

Autres considérations

La liste suivante contient d'autres considérations relatives à EMR Serverless.

- EMR Serverless est disponible dans les versions suivantes : Régions AWS
  - USA Est (Ohio)
  - USA Est (Virginie du Nord)
  - USA Ouest (Californie du Nord)
  - USA Ouest (Oregon)
  - Afrique (Le Cap)
  - Asie-Pacifique (Hong Kong)
  - Asie-Pacifique (Jakarta)
  - Asie-Pacifique (Mumbai)
  - Asie-Pacifique (Osaka)
  - Asia Pacific (Seoul)
  - Asie-Pacifique (Singapour)
  - Asie-Pacifique (Sydney)
  - Asie-Pacifique (Tokyo)
  - Canada (Centre)
  - Europe (Francfort)
  - Europe (Irlande)
  - Europe (Londres)
  - Europe (Milan)
  - Europe (Paris)
  - Europe (Espagne)
  - Europe (Stockholm)
  - Middle East (Bahrain)
  - Moyen-Orient (EAU)
  - Amérique du Sud (São Paulo)
  - AWS GovCloud (USA Est)
  - AWS GovCloud (US-Ouest)

Pour obtenir la liste des points de terminaison associés à ces régions, reportez-vous àPoints de terminaison de service.

- Le délai d'expiration par défaut pour l'exécution d'une tâche est de 12 heures. Vous pouvez modifier ce paramètre à l'aide de la `executionTimeoutMinutes` propriété dans l'`startJobRun`API ou le AWS SDK. Vous pouvez `executionTimeoutMinutes` définir la valeur 0 si vous souhaitez que l'exécution de votre tâche ne s'arrête jamais. Par exemple, si vous avez une application de streaming, définissez la valeur 0 `executionTimeoutMinutes` pour permettre à la tâche de streaming de s'exécuter en continu.
- La `billedResourceUtilization` propriété de l'`getJobRun`API indique le vCPU, la mémoire et le stockage agrégés facturés pour l'exécution du job. AWS Les ressources facturées incluent une utilisation minimale d'une minute pour les employés, ainsi qu'un stockage supplémentaire de plus de 20 Go par travailleur. Ces ressources n'incluent pas l'utilisation pour les travailleurs préinitialisés inactifs.
- Sans connectivité VPC, une tâche peut accéder à certains Service AWS points de terminaison de celui-ci. Région AWS Ces services incluent Amazon S3, AWS Glue, AWS Lake Formation, Amazon CloudWatch Logs AWS KMS, AWS Security Token Service, Amazon DynamoDB, et AWS Secrets Manager. Vous pouvez activer la connectivité VPC pour accéder à un autre Services AWS canal AWS PrivateLink, mais cela n'est pas obligatoire. Pour accéder à des services externes, créez votre application avec un VPC.
- EMR Serverless ne prend pas en charge le HDFS. Les disques locaux situés sur les travailleurs constituent un stockage temporaire qu'EMR Serverless utilise pour mélanger et traiter les données lors de l'exécution des tâches.

Versions de lancement d'Amazon EMR Serverless

Une version d'Amazon EMR est un ensemble d'applications open source issues de l'écosystème des mégadonnées. Chaque version inclut des applications, des composants et des fonctionnalités Big Data que vous sélectionnez pour qu'Amazon EMR Serverless déploie et configure lors de l'exécution de votre tâche.

Avec Amazon EMR 6.6.0 et versions ultérieures, déployez EMR Serverless. Cette option de déploiement n'est pas disponible avec les versions antérieures d'Amazon EMR. Lorsque vous soumettez votre travail, spécifiez l'une des versions prises en charge suivantes.

Rubriques

- EMR Serverless6,10
- EMR Serverless6,9.0
- EMR Serverless6,8.0
- EMR Serverless6.7.0
- EMR Serverless6.6.0

AWS runtime for Apache Spark(aperçu d'emr-spark-8.0)

Le tableau suivant répertorie les versions de l'application disponibles avec `AWS runtime for Apache Spark` (emr-spark-8.0-preview).

Informations sur la version de l'application

Application	Version
Spark	4.0.1-amzn-0

`AWS runtime for Apache Spark`notes de mise à jour (emr-spark-8.0-preview)

- Version préliminaire — Il s'agit d'une version préliminaire d'`AWS runtime for Apache Spark`Apache Spark 4.0.1. Cette version préliminaire est uniquement disponible sur EMR Serverless.
- Disponibilité régionale - Cette version préliminaire est disponible dans toutes les AWS régions où EMR Serverless est disponible, à l'exception de la Chine et des AWS GovCloud États-Unis.
- Informations sur la version de l'application - Cette version est fournie avec les versions d'application suivantes :
  - AWS SDK pour Java 2.35.5, 1.12.792
  - Python 3.9, 3.11, 3.12
  - Scala 2.13.16
  - AmazonCloudWatchAgent 1.300034.0-amzn-0
  - Delta 4.0.0-amzn-0-spark
  - Iceberg 1.10.0-amzn-spark-0
  - Cette version est fournie par défaut avec Amazon Corretto 17 (basé sur OpenJDK) pour les applications compatibles avec Corretto 17 (JDK 17).

- Limites de la version préliminaire - Les fonctionnalités suivantes ne sont pas disponibles dans cette version préliminaire :
  - Fonctionnalités interactives et d'intégration : SageMaker Unified Studio, intégration d'EMR Studio, Spark Connect, Livy, etc. JupyterEnterpriseGateway ne sont pas prises en charge.
  - Formats de table et contrôle d'accès : Hudi, Delta Universal Format et le contrôle d'accès détaillé (FGAC) avec filtrage et opérateurs au niveau des lignes ou des colonnes ne sont pas pris en charge. DDL/DML
  - Connecteurs de données : spark-sql-kinesis les connecteurs emr-dynamodb et spark-redshift ne sont pas disponibles.
  - Serveur d'historique : le serveur d'historique Spark persistant n'est pas disponible dans cette version préliminaire. Les utilisateurs peuvent toujours accéder à l'interface utilisateur en direct de Spark pour surveiller et déboguer les tâches sans serveur actives en temps réel.
  - Fonctionnalités spécialisées : Les vues matérialisées ne sont pas disponibles.
- Fonctionnalités de prévisualisation : vous pouvez tester les fonctionnalités suivantes dans cette version préliminaire. Cette version préliminaire n'est pas recommandée pour les charges de travail de production :
  - Fonctionnalités SQL : mode ANSI SQL avec une gestion des types plus stricte, syntaxe SQL PIPE (|>) pour les opérations de chaînage, type de données VARIANT pour les données JSON semi-structurées, scripts SQL avec instructions de flux de contrôle et variables de session, et fonctions SQL définies par l'utilisateur.
  - Améliorations du streaming : API de traitement statique arbitraire v2 avec transformWithState opérateur, lecteur de source de données d'état pour un état de streaming interrogeable (expérimental) et magasin d'état amélioré avec point de contrôle amélioré du journal des modifications RockSDB.
  - Support du format de tableau : Apache Iceberg v3 avec prise en charge des types de données VARIANT, intégration des tables AWS S3 et accès complet aux tables (FTA) AWS Lake Formation pour les tables Iceberg, Delta Lake et Hive.
- Documentation supplémentaire - Pour de la documentation supplémentaire sur Apache Spark, consultez la documentation de la version 4.0.1 d'Apache Spark.

Démarrage

Pour commencer à utiliser la version préliminaire d'Apache Spark 4.0.1, créez une application EMR Serverless à l'aide de la CLI : AWS

```
aws emr-serverless create-application --type spark \
  --release-label emr-spark-8.0-preview \
  --region us-east-1 --name spark4-preview
```

EMR Serverless7,12,0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.12.0.

Application	Version
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 7.12.0

- Nouvelles fonctionnalités
  - Stockage sans serveur pour EMR Serverless — Amazon EMR Serverless introduit le stockage sans serveur, avec EMR version 7.12 et versions ultérieures, qui élimine le provisionnement de disques locaux pour les charges de travail Apache Spark. EMR Serverless gère automatiquement les opérations de données intermédiaires telles que le shuffle, sans frais de stockage. Le stockage sans serveur dissocie le stockage du calcul, ce qui permet à Spark de libérer les employés immédiatement lorsqu'ils sont inactifs plutôt que de les maintenir actifs pour préserver des données temporaires. Pour en savoir plus, veuillez consulter la section aws.amazon.com/serverless-storage-for-emr-serverless.
  - Iceberg Materialized Views - À partir d'Amazon EMR 7.12.0, Amazon EMR Spark prend en charge la création et la gestion des Iceberg Materialized Views (MV)
  - Accès complet aux tables à Hudi : depuis Amazon EMR 7.12.0, Amazon EMR prend désormais en charge le contrôle de l'accès complet aux tables (FTA) 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 d'Iceberg - Amazon EMR 7.12.0 prend en charge la version 1.10 d'Apache Iceberg

EMR Serverless7,11.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.11.0.

Application	Version
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 7.11.0

- Durée maximale d'exécution des tâches : la durée maximale `StartJobRun` d'exécution des tâches BATCH est de 7 jours à compter de cette version. `executionTimeoutMinutes` `executionTimeoutMinutes`ne peut plus être réglé sur aucun `0` délai d'attente, pour les exécutions par lots.

EMR Serverless7,1,0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.10.0.

Application	Version
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 7.10.0

- Métriques pour EMR sans serveur : les métriques de surveillance sont restructurées pour se concentrer sur les dimensions et. `ApplicationName` `JobName` Les anciens indicateurs ne seront plus mis à jour à l'avenir. Pour plus d'informations, reportez-vous à la section Surveillance des applications et des tâches EMR sans serveur.

EMR Serverless7.9.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.9.0.

Application	Version
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7,8.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.8.0.

Application	Version
Apache Spark	3.5.4
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7.7.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.7.0.

Application	Version
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7.6.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.6.0.

Application	Version
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7.5.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.5.0.

Application	Version
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7.4.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.4.0.

Application	Version
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7.3.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.3.0.

Application	Version
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 7.3.0

- Simultanéité des tâches et mise en file d'attente avec EMR Serverless : la simultanéité des tâches et la mise en file d'attente sont activées par défaut lorsque vous créez une nouvelle application EMR sans serveur sur Amazon EMR version 7.3.0 ou ultérieure. Pour plus d'informations, reportez-vous àthe section called “Simultanéité des tâches et mise en file d'attente”, qui explique comment démarrer avec la simultanéité et la mise en file d'attente et contient également une liste de considérations relatives aux fonctionnalités.

EMR Serverless7.2.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.2.0.

Application	Version
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 7.2.0

- Lake Formation with EMR Serverless : vous pouvez désormais appliquer des contrôles AWS Lake Formation d'accès précis aux tables du catalogue de données soutenues par 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 vos tâches EMR Serverless Spark. Pour plus d'informations, consultez the section called "Lake Formation pour FGAC" et the section called "Considérations et restrictions".

EMR Serverless7.1.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.1.0.

Application	Version
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless7.0.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 7.0.0.

Application	Version
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless6,15,0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.15.0.

Application	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 6.15.0

- Support TLS : avec les versions 6.15.0 et supérieures d'Amazon EMR Serverless, activez les communications cryptées par protocole TLS mutuel entre les collaborateurs lors de vos exécutions de tâches Spark. Lorsqu'il est activé, EMR Serverless génère automatiquement un certificat unique pour chaque travailleur, qu'il fournit dans le cadre d'une exécution de tâches que les travailleurs utilisent lors de la prise de contact TLS pour s'authentifier mutuellement et établir un canal crypté afin de traiter les données en toute sécurité. Pour plus d'informations sur le chiffrement TLS mutuel, reportez-vous à la section Chiffrement inter-opérateurs.

EMR Serverless6,14.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.14.0.

Application	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless6,13.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.13.0.

Application	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0,1,2

EMR Serverless6,12,0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.12.0.

Application	Version
Apache Spark	3.4.0
Apache Hive	3.1.3

Application	Version
Apache Tez	0,1,2

EMR Serverless6,11,0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.11.0.

Application	Version
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 6.11.0

- Accès aux ressources S3 depuis d'autres comptes : avec les versions 6.11.0 et supérieures, vous pouvez configurer plusieurs rôles IAM à assumer lorsque vous accédez à des compartiments Amazon S3 dans différents comptes AWS depuis EMR Serverless.

EMR Serverless6,10

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.10.0.

Application	Version
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 6.10.0

- Pour les applications EMR Serverless de version 6.10.0 ou supérieure, la valeur par défaut de la propriété est. `spark.dynamicAllocation.maxExecutors` `infinity` Les versions antérieures sont définies par défaut sur`100`. Pour plus d'informations, consultez Propriétés de la tâche Spark.

EMR Serverless6,9.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.9.0.

Application	Version
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0,1,2

Notes de mise à jour d'EMR Serverless 6.9.0

- 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 Serverless.
- La version 6.9.0 d'EMR Serverless ajoute la prise en charge de l'architecture AWS Graviton2 (arm64). Vous pouvez utiliser le `architecture` paramètre pour `create-application` et pour `update-application` APIs choisir l'architecture arm64. Pour plus d'informations, consultez Options d'architecture sans serveur Amazon EMR.
- Vous pouvez désormais exporter, importer, interroger et joindre des tables Amazon DynamoDB directement depuis vos applications EMR Serverless Spark et Hive. Pour plus d'informations, consultez Connexion à DynamoDB avec Amazon EMR Serverless.

Problèmes connus

- 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 Serverless6,8.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.8.0.

Application	Version
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless6.7.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.7.0.

Application	Version
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

Modifications, améliorations et problèmes résolus spécifiques au moteur

Le tableau suivant répertorie une nouvelle fonctionnalité spécifique au moteur.

Modifier	Description
Fonctionnalité	Le planificateur Tez prend désormais en charge la préemption de la tâche Tez au lieu de la préemption du conteneur

EMR Serverless6.6.0

Le tableau suivant répertorie les versions de l'application disponibles avec la version EMR Serverless 6.6.0.

Application	Version
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

Notes de mise à jour initiales d'EMR Serverless

- EMR Serverless prend en charge la classification de configuration Spark. `spark-defaults` Cette classification modifie les valeurs du fichier `spark-defaults.conf` XML de Spark. Les classifications de configuration vous permettent de personnaliser les applications. Pour plus d'informations, reportez-vous à la section Configuration des applications.
- EMR Serverless prend en charge les classifications de configuration Hive`hive-site`,, et`tez-site`. `emrfs-site` `core-site` Cette classification peut modifier les valeurs du fichier Hive, du `hive-site.xml` `tez-site.xml` fichier de Tez, des paramètres EMRFS d'Amazon EMR ou du fichier Hadoop, respectivement. `core-site.xml` Les classifications de configuration vous permettent de personnaliser les applications. Pour plus d'informations, reportez-vous à la section Configuration des applications.

Modifications, améliorations et problèmes résolus spécifiques au moteur

- Le tableau suivant répertorie les rétroportages Hive et Tez.

Changements liés à Hive et Tez

Modifier	Description
Rétroportage	TEZ-4430 : Correction d'un problème de propriété `tez.task.launch.cmd-opts`
Rétroportage	HIVE-25971 : Correction des retards d'arrêt de la tâche Tez dus à l'ouverture d'un pool de threads en cache

Historique du document

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version d'EMR Serverless. Pour plus d'informations sur les mises à jour de cette documentation, vous pouvez vous abonner à un flux RSS.

Modification	Description	Date
Nouvelle version préliminaire	`AWS runtime for Apache Spark`(aperçu d'emr-spark-8.0)	21 novembre 2025
Nouvelle version	EMR Serverless7.2.0	25 juillet 2024
Nouvelle version	EMR Serverless7.1.0	17 avril 2024
Mise à jour d'une politique existante.	Ajout du nouveau `Sid` `CloudWatchPolicyStatement` et `EC2PolicyStatement` à la EMRServerlessServiceRolePolicy politique d'Amazon.	25 janvier 2024
Nouvelle version	EMR Serverless7.0.0	29 décembre 2023
Nouvelle version	EMR Serverless6,15,0	17 novembre 2023
Nouvelle fonction	Configurez plusieurs rôles IAM à assumer lorsque vous accédez à des compartiments Amazon S3 dans différents comptes depuis EMR Serverless (6.11 et versions ultérieures)	18 octobre 2023
Nouvelle version	EMR Serverless6,14.0	17 octobre 2023

Nouvelle fonction	Configuration d'application par défaut pour EMR Serverless	25 septembre 2023
Mise à jour des propriétés Hive par défaut	Mise à jour des valeurs par défaut pour les propriétés des tâches `hive.driver.disk` `hive.tez.disk.size` `hive.tez.auto.reducer.parallelism` ,, et `tez.grouping.min-size` Hive.	12 septembre 2023
Nouvelle version	EMR Serverless6,13.0	11 septembre 2023
Nouvelle version	EMR Serverless6,12,0	21 juillet 2023
Nouvelle version	EMR Serverless6,11,0	8 juin 2023
Mise à jour de la politique des rôles liés aux services	Le rôle `AmazonEMRServerlessServiceRolePolicy` SLR a été mis à jour pour publier l'utilisation au niveau du compte dans l'espace de noms. `"AWS/Usage"`	20 avril 2023
EMR Serverlessdisponibilité générale (GA)	Il s'agit de la première version publique d'EMR Serverless.	1er juin 2022

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.