Guia do usuário EMR do Amazon Serverless

Amazon EMR

Amazon EMR: Guia do usuário EMR do Amazon Serverless

Table of Contents

O que é Amazon EMR Serverless?

O Amazon EMR Serverless é uma opção de implantação para a Amazon EMR que fornece um ambiente de execução sem servidor. Isso simplifica a operação de aplicativos de análise que usam as estruturas de código aberto mais recentes, como o Apache Spark e o Apache Hive. Com o EMR Serverless, você não precisa configurar, otimizar, proteger ou operar clusters para executar aplicativos com essas estruturas.

EMRO Serverless ajuda você a evitar o provisionamento excessivo ou insuficiente de recursos para seus trabalhos de processamento de dados. EMRO Serverless determina automaticamente os recursos de que o aplicativo precisa, obtém esses recursos para processar seus trabalhos e os libera quando os trabalhos são concluídos. Para casos de uso em que os aplicativos precisam de uma resposta em segundos, como análise interativa de dados, você pode pré-inicializar os recursos de que o aplicativo precisa ao criar o aplicativo.

Com o EMR Serverless, você continuará obtendo os benefícios da AmazonEMR, como compatibilidade de código aberto, simultaneidade e desempenho otimizado de tempo de execução para estruturas populares.

EMRO Serverless é adequado para clientes que desejam facilidade na operação de aplicativos usando estruturas de código aberto. Ele oferece inicialização rápida de trabalhos, gerenciamento automático de capacidade e controles de custos diretos.

Conceitos

Nesta seção, abordamos os termos e conceitos de EMR Serverless que aparecem em nosso Guia do Usuário EMR Serverless.

Versão de lançamento

Uma EMR versão da Amazon é um conjunto de aplicativos de código aberto do ecossistema de big data. Cada versão inclui diferentes aplicativos, componentes e recursos de big data que você seleciona para que o EMR Serverless implante e configure para que eles possam executar seus aplicativos. Ao criar um aplicativo, você deve especificar sua versão de lançamento. Escolha a versão de EMR lançamento da Amazon e a versão da estrutura de código aberto que você deseja usar em seu aplicativo. Para saber mais sobre as versões de pré-lançamento, consulteVersões de EMR lançamento do Amazon Serverless.

Aplicativo

Com o EMR Serverless, você pode criar um ou mais aplicativos EMR sem servidor que usam estruturas de análise de código aberto. Para criar um aplicativo, você deve especificar os seguintes atributos:

- A versão de EMR lançamento da Amazon para a versão da estrutura de código aberto que você deseja usar. Para determinar sua versão de lançamento, consulteVersões de EMR lançamento do Amazon Serverless.
- O tempo de execução específico que você deseja que seu aplicativo use, como o Apache Spark ou o Apache Hive.

Depois de criar um aplicativo, você pode enviar trabalhos de processamento de dados ou solicitações interativas ao seu aplicativo.

Cada aplicativo EMR sem servidor é executado em uma Amazon Virtual Private Cloud (VPC) segura, estritamente separada de outros aplicativos. Além disso, você pode usar AWS Identity and Access Management (IAM) políticas para definir quais usuários e funções podem acessar o aplicativo. Você também pode especificar limites para controlar e rastrear os custos de uso incorridos pelo aplicativo.

Considere criar vários aplicativos quando precisar fazer o seguinte:

- Use diferentes estruturas de código aberto
- Use versões diferentes de estruturas de código aberto para diferentes casos de uso
- Execute testes A/B ao atualizar de uma versão para outra
- Mantenha ambientes lógicos separados para cenários de teste e produção
- Forneça ambientes lógicos separados para equipes diferentes com controles de custos e rastreamento de uso independentes
- Separe line-of-business aplicativos diferentes

EMRO Serverless é um serviço regional que simplifica a forma como as cargas de trabalho são executadas em várias zonas de disponibilidade em uma região. Para saber mais sobre como usar aplicativos com o EMR Serverless, consulte. Interagindo com um aplicativo

Execução de trabalho

A execução de um trabalho é uma solicitação enviada a um aplicativo EMR sem servidor que o aplicativo executa e acompanha de forma assíncrona até a conclusão. Exemplos de trabalhos incluem uma consulta HiveQL que você envia para um aplicativo Apache Hive ou um script de processamento de dados que você envia para PySpark um aplicativo Apache Spark. Ao enviar um trabalho, você deve especificar uma função de tempo de execução, de autoria emIAM, que o trabalho usa para acessar AWS recursos, como objetos do Amazon S3. Você pode enviar várias solicitações de execução de trabalho para um aplicativo, e cada execução de trabalho pode usar uma função de tempo de execução diferente para acessar AWS recursos. Um aplicativo EMR sem servidor começa a executar trabalhos assim que os recebe e executa várias solicitações de trabalho simultaneamente. Para saber mais sobre como o EMR Serverless executa trabalhos, consulte. Execução de trabalhos

Operadores

Um aplicativo EMR sem servidor usa internamente trabalhadores para executar suas cargas de trabalho. Os tamanhos padrão desses trabalhadores são baseados no tipo de aplicativo e na versão de EMR lançamento da Amazon. Ao programar a execução de um trabalho, você pode substituir esses tamanhos.

Quando você envia um trabalho, o EMR Serverless calcula os recursos que o aplicativo precisa para o trabalho e agenda os trabalhadores. EMRO Serverless divide suas cargas de trabalho em tarefas, baixa imagens, provisiona e configura funcionários e os descomissiona quando o trabalho é concluído. EMRO Serverless aumenta ou diminui automaticamente os funcionários com base na carga de trabalho e no paralelismo necessários em cada estágio do trabalho. Esse escalonamento automático elimina a necessidade de estimar o número de trabalhadores que o aplicativo precisa para executar suas cargas de trabalho.

Capacidade pré-inicializada

EMRO Serverless fornece um recurso de capacidade pré-inicializado que mantém os trabalhadores inicializados e prontos para responder em segundos. Essa capacidade cria efetivamente um grupo caloroso de trabalhadores para um aplicativo. Para configurar esse recurso para cada aplicativo, defina o `initial-capacity` parâmetro de um aplicativo. Quando você configura a capacidade pré-inicializada, os trabalhos podem começar imediatamente para que você possa implementar aplicativos iterativos e trabalhos urgentes. Para saber mais sobre trabalhadores pré-inicializados, consulte. Configurando um aplicativo

EMREstúdio

EMRO Studio é o console do usuário que você pode usar para gerenciar seus aplicativos EMR sem servidor. Se não existir um EMR Studio em sua conta quando você criar seu primeiro aplicativo EMR sem servidor, criaremos automaticamente um para você. Você pode acessar o EMR Studio pelo EMR console da Amazon ou ativar o acesso federado do seu provedor de identidade (IdP) IAM por meio IAM do Identity Center. Ao fazer isso, os usuários podem acessar o Studio e gerenciar aplicativos sem EMR servidor sem acesso direto ao console da AmazonEMR. Para saber mais sobre como os aplicativos EMR sem servidor funcionam com o EMR Studio, consulte e. Interagindo com seu aplicativo a partir do console do EMR Studio Executando trabalhos a partir do console do EMR Studio

Pré-requisitos para começar a usar o Serverless EMR

Tópicos

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar uma.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e inserir um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário root tem acesso a todos Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilitar AWS IAM Identity Center e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Consolecomo proprietário da conta, escolhendo o usuário root e inserindo seu Conta da AWS endereço de e-mail. Na próxima página, insira sua senha.

   Para obter ajuda para fazer login usando o usuário root, consulte Como fazer login como usuário root no Início de Sessão da AWS Guia do usuário.
2. Ative a autenticação multifator (MFA) para seu usuário root.

   Para obter instruções, consulte Habilitar um MFA dispositivo virtual para seu Conta da AWS usuário root (console) no Guia do IAM usuário.

Criar um usuário com acesso administrativo

1. Ative o IAM Identity Center.

   Para obter instruções, consulte Habilitando AWS IAM Identity Center no AWS IAM Identity Center Guia do usuário.
2. No IAM Identity Center, conceda acesso administrativo a um usuário.

   Para um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como sua fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no AWS IAM Identity Center Guia do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para entrar com seu usuário do IAM Identity Center, use o login URL que foi enviado ao seu endereço de e-mail quando você criou o usuário do IAM Identity Center.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no AWS portal de acesso no Início de Sessão da AWS Guia do usuário.

Atribuir acesso a usuários adicionais

1. No IAM Identity Center, crie um conjunto de permissões que siga as melhores práticas de aplicação de permissões com privilégios mínimos.

Para obter instruções, consulte Criar um conjunto de permissões no AWS IAM Identity Center Guia do usuário.

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Adicionar grupos no AWS IAM Identity Center Guia do usuário.

Conceder permissões

Em ambientes de produção, recomendamos que você use políticas mais refinadas. Para obter exemplos dessas políticas, consulteExemplos de políticas de acesso de usuários para EMR Serverless. Para saber mais sobre gerenciamento de acesso, consulte Gerenciamento de acesso para AWS recursos no Guia do IAM usuário.

Para usuários que precisam começar a usar o EMR Serverless em um ambiente sandbox, use uma política semelhante à seguinte:

```
{
    "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/*"
        }
    ]
}
```

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criar um conjunto de permissões no AWS IAM Identity Center Guia do usuário.

- Usuários gerenciados IAM por meio de um provedor de identidade:

  Crie um perfil para a federação de identidades. Siga as instruções em Criação de uma função para um provedor de identidade terceirizado (federação) no Guia IAM do usuário.

- IAMusuários:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de uma função para um IAM usuário no Guia IAM do usuário.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adicionar permissões a um usuário (console) no Guia do IAM usuário.

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no IAM Identity Center)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou AWS APIs.	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para usar AWS IAM Identity Center no AWS Command Line Interface Guia do usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte Autenticação do IAM Identity Center no AWS SDKsGuia de referência de ferramentas e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou AWS APIs.	Seguindo as instruções em Usando credenciais temporárias com AWS recursos no Guia do IAM usuário.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para o	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Autenticação usando

Qual usuário precisa de acesso programático?	Para	Por
	AWS CLI, AWS SDKs, ou AWS APIs.	credenciais de IAM usuário no AWS Command Line Interface Guia do usuário. • Para AWS SDKse ferrament as, consulte Autentica r usando credenciais de longo prazo no AWS SDKsGuia de referência de ferramentas e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para IAM usuários no Guia IAM do usuário.

Instale e configure o AWS CLI

Se você quiser usar o EMR ServerlessAPIs, deverá instalar a versão mais recente do AWS Command Line Interface (AWS CLI). Você não precisa do AWS CLI para usar o EMR Serverless no console do EMR Studio, e você pode começar sem o CLI seguindo as etapas em. Introdução ao EMR Serverless a partir do console

Para configurar o AWS CLI

1. Para instalar a versão mais recente do AWS CLI para macOS, Linux ou Windows, consulte Instalando ou atualizando a versão mais recente do AWS CLI.
2. Para configurar o AWS CLI e configuração segura do seu acesso a Serviços da AWS, incluindo EMR Serverless, consulte Configuração rápida com. `aws configure`
3. Para verificar a configuração, digite o seguinte DataBrew comando no prompt de comando.

```
aws emr-serverless help
```

AWS CLI os comandos usam o padrão Região da AWS da sua configuração, a menos que você a defina com um parâmetro ou um perfil. Para definir seu Região da AWS com um parâmetro, você pode adicionar o `--region` parâmetro a cada comando.

Para definir seu Região da AWS com um perfil, primeiro adicione um perfil nomeado no `~/.aws/config` arquivo ou no `%UserProfile%/.aws/config` arquivo (para Microsoft Windows). Siga as etapas em Perfis nomeados para o AWS CLI. Em seguida, defina seu Região da AWS e outras configurações com um comando semelhante ao do exemplo a seguir.

```
[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
```

Abra o console do

A maioria dos tópicos orientados ao console nesta seção começa no console da Amazon. EMR Se você ainda não estiver conectado ao seu Conta da AWS, faça login, abra o EMRconsole da Amazon e vá para a próxima seção para continuar começando a usar a AmazonEMR.

Começando a usar o Amazon EMR Serverless

Este tutorial ajuda você a começar a usar o EMR Serverless ao implantar uma amostra de carga de trabalho do Spark ou do Hive. Você criará, executará e depurará seu próprio aplicativo. Mostramos as opções padrão na maior parte deste tutorial.

Antes de iniciar um aplicativo EMR sem servidor, conclua as tarefas a seguir.

Tópicos

Conceda permissões para usar o EMR Serverless

Para usar o EMR Serverless, você precisa de um usuário ou IAM função com uma política anexada que conceda permissões para EMR o Serverless. Para criar um usuário e anexar a política apropriada a esse usuário, siga as instruções emConceder permissões.

Prepare o armazenamento sem EMR servidor

Neste tutorial, você usará um bucket do S3 para armazenar arquivos e registros de saída da amostra de carga de trabalho do Spark ou do Hive que você executará usando um aplicativo sem servidor. EMR Para criar um bucket, siga as instruções em Criar um bucket no Guia do usuário do console do Amazon Simple Storage Service. Substitua qualquer referência adicional *DOC-EXAMPLE-BUCKET* a pelo nome do bucket recém-criado.

Crie um EMR estúdio para executar cargas de trabalho interativas

Se você quiser usar o EMR Serverless para executar consultas interativas por meio de notebooks hospedados no EMR Studio, você precisa especificar um bucket S3 e a função de serviço

mínima para o EMR Serverless criar um espaço de trabalho. Para ver as etapas de configuração, consulte Configurar um EMR estúdio no Guia EMR de gerenciamento da Amazon. Para obter mais informações sobre cargas de trabalho interativas, consulteExecute cargas de trabalho interativas com o EMR Serverless por meio do Studio EMR.

Crie uma função de tempo de execução do trabalho

Os trabalhos executados no EMR Serverless usam uma função de tempo de execução que fornece permissões granulares para tarefas específicas Serviços da AWS e recursos em tempo de execução. Neste tutorial, um bucket público do S3 hospeda os dados e os scripts. O bucket *DOC-EXAMPLE-BUCKET* armazena a saída.

Para configurar uma função de tempo de execução do trabalho, primeiro crie uma função de tempo de execução com uma política de confiança para que o EMR Serverless possa usar a nova função. Em seguida, anexe a política de acesso do S3 necessária a essa função. As etapas a seguir orientam você pelo processo.

Console

1. Navegue até o console do IAM no https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Roles.
3. Selecione Criar função.
4. Para o tipo de função, escolha Política de confiança personalizada e cole a política de confiança a seguir. Isso permite que trabalhos enviados para seus aplicativos Amazon EMR Serverless acessem outros Serviços da AWS em seu nome.

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

5. Escolha Avançar para navegar até a página Adicionar permissões e, em seguida, escolha Criar política.
6. A página Criar política é aberta em uma nova guia. Cole a política JSON abaixo.

> **Important**
>
> *DOC-EXAMPLE-BUCKET*Substitua a política abaixo pelo nome real do bucket criado emPrepare o armazenamento sem EMR servidor. Essa é uma política básica para acesso ao S3. Para obter mais exemplos de funções de execução de tarefas, consulteFunções de tempo de execução de trabalho para Amazon EMR Serverless.

```
{
    "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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-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": ["*"]
        }
    ]
}
```

7. Na página Revisar política, insira um nome para sua política, como`EMRServerlessS3AndGlueAccessPolicy`.
8. Atualize a página Anexar política de permissões e escolha`EMRServerlessS3AndGlueAccessPolicy`.
9. Na página Nome, revisão e criação, em Nome da função, insira um nome para sua função, por exemplo,`EMRServerlessS3RuntimeRole`. Para criar essa IAM função, escolha Criar função.

CLI

1. Crie um arquivo chamado `emr-serverless-trust-policy.json` que contenha a política de confiança a ser usada para a IAM função. O arquivo deve conter a seguinte política.

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

```
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
            "Service": "emr-serverless.amazonaws.com"
        }
    }]
}
```

2. Crie uma IAM função chamadaEMRServerlessS3RuntimeRole. Use a política de confiança que você criou na etapa anterior.

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

Observe ARN na saída. Você usa a ARN da nova função durante o envio do trabalho, depois chamada de*job-role-arn*.

3. Crie um arquivo chamado emr-sample-access-policy.json que defina a IAM política para sua carga de trabalho. Isso fornece acesso de leitura ao script e aos dados armazenados em buckets públicos do S3 e acesso de leitura e gravação a. *DOC-EXAMPLE-BUCKET*

> **Important**
>
> Substitua *DOC-EXAMPLE-BUCKET* na política abaixo pelo nome real do bucket criado emPrepare o armazenamento sem EMR servidor.. Esta é uma política básica para AWS Glue e acesso ao S3. Para obter mais exemplos de funções de execução de tarefas, consulteFunções de tempo de execução de trabalho para Amazon EMR Serverless.

```
{
    "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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",Understanding default application behavior,
including auto-start and auto-stop, as well as maximum capacity and worker
configurations for configuring an application with &EMRServerless;.
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
```

```
}
```

4. Crie uma IAM política nomeada `EMRServerlessS3AndGlueAccessPolicy` com o arquivo de política que você criou na Etapa 3. Anote o ARN resultado, pois você usará ARN a nova política na próxima etapa.

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

Observe a nova política ARN na saída. Você o substituirá *`policy-arn`* na próxima etapa.

5. Anexe a IAM política `EMRServerlessS3AndGlueAccessPolicy` à função de tempo de execução do trabalho`EMRServerlessS3RuntimeRole`.

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

Introdução ao EMR Serverless a partir do console

Etapas a serem executadas

- Etapa 1: criar um aplicativo EMR sem servidor
- Etapa 2: Enviar uma execução de trabalho ou carga de trabalho interativa
- Etapa 3: Exibir a interface do usuário e os registros do aplicativo
- Etapa 4: limpar

Etapa 1: criar um aplicativo EMR sem servidor

Crie um novo aplicativo com o EMR Serverless da seguinte maneira.

1. Faça login no AWS Management Console e abra o EMR console da Amazon em https://console.aws.amazon.com/emr.
2. No painel de navegação esquerdo, escolha EMRSem servidor para navegar até a página inicial sem EMR servidor.
3. Para criar ou gerenciar aplicativos EMR sem servidor, você precisa da interface do usuário do EMR Studio.

- Se você já tem um EMR estúdio no Região da AWS onde você deseja criar um aplicativo e, em seguida, selecione Gerenciar aplicativos para navegar até o seu EMR Studio ou selecione o estúdio que você deseja usar.
- Se você não tiver um EMR estúdio no Região da AWS onde você quiser criar um aplicativo, escolha Começar e, em seguida, escolha Criar e iniciar o Studio. EMRO Serverless cria um EMR Studio para você, para que você possa criar e gerenciar aplicativos.

4. Na interface do usuário do Create Studio que se abre em uma nova guia, insira o nome, o tipo e a versão de lançamento do seu aplicativo. Se você quiser executar somente trabalhos em lotes, selecione Usar configurações padrão somente para trabalhos em lotes. Para cargas de trabalho interativas, selecione Usar configurações padrão para cargas de trabalho interativas. Você também pode executar trabalhos em lotes em aplicativos interativos com essa opção. Se precisar, você pode alterar essas configurações posteriormente.

   Para obter mais informações, consulte Criar um estúdio.
5. Selecione Criar aplicativo para criar seu primeiro aplicativo.

Continue na próxima seção Etapa 2: Enviar uma execução de trabalho ou carga de trabalho interativa para enviar uma execução de trabalho ou uma carga de trabalho interativa.

Etapa 2: Enviar uma execução de trabalho ou carga de trabalho interativa

Spark job run

Neste tutorial, usamos um PySpark script para calcular o número de ocorrências de palavras únicas em vários arquivos de texto. Um bucket S3 público e somente para leitura armazena o script e o conjunto de dados.

Para executar uma tarefa do Spark

1. Faça o upload do script de amostra `wordcount.py` em seu novo bucket com o comando a seguir.

   ```
   aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
   scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
   ```
2. Etapa 1: criar um aplicativo EMR sem servidorA conclusão leva você à página de detalhes do aplicativo no EMR Studio. Lá, escolha a opção Enviar trabalho.

3. Na página Enviar trabalho, conclua o seguinte.

   - No campo Nome, insira o nome que você deseja chamar de execução do trabalho.
   - No campo Função de tempo de execução, insira o nome da função que você criou emCrie uma função de tempo de execução do trabalho.
   - No campo Localização do script, insira `s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py` como S3URI.
   - No campo Argumentos do script, insira`["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/output"]`.
   - Na seção Propriedades do Spark, escolha Editar como texto e insira as seguintes configurações.

   ```
   --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. Para iniciar a execução do trabalho, escolha Enviar trabalho.
5. Na guia Job runs, você deve ver seu novo job sendo executado com o status Running.

Hive job run

Nesta parte do tutorial, criamos uma tabela, inserimos alguns registros e executamos uma consulta de agregação de contagem. Para executar o trabalho do Hive, primeiro crie um arquivo que contenha todas as consultas do Hive a serem executadas como parte de um único trabalho, carregue o arquivo no S3 e especifique esse caminho do S3 ao iniciar o trabalho do Hive.

Para executar um trabalho do Hive

1. Crie um arquivo chamado `hive-query.ql` que contenha todas as consultas que você deseja executar em seu trabalho do 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. Faça `hive-query.ql` o upload para seu bucket do S3 com o comando a seguir.

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql
```

3. Etapa 1: criar um aplicativo EMR sem servidorA conclusão leva você à página de detalhes do aplicativo no EMR Studio. Lá, escolha a opção Enviar trabalho.
4. Na página Enviar trabalho, conclua o seguinte.
   - No campo Nome, insira o nome que você deseja chamar de execução do trabalho.
   - No campo Função de tempo de execução, insira o nome da função que você criou emCrie uma função de tempo de execução do trabalho.
   - No campo Localização do script, insira `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql` como S3URI.
   - Na seção Propriedades do Hive, escolha Editar como texto e insira as seguintes configurações.

     ```
     --hiveconf hive.log.explain.output=false
     ```
   - Na seção Configuração do Job, escolha Editar como JSON e digite o seguinteJSON.

     ```
     {
        "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"
               }
         }]
     }
     ```
5. Para iniciar a execução do trabalho, escolha Enviar trabalho.
6. Na guia Job runs, você deve ver seu novo job sendo executado com o status Running.

Interactive workload

Com o Amazon EMR 6.14.0 e versões posteriores, você pode usar notebooks hospedados no EMR Studio para executar cargas de trabalho interativas para o Spark no Serverless. EMR Para obter mais informações, incluindo permissões e pré-requisitos, consulte. Execute cargas de trabalho interativas com o EMR Serverless por meio do Studio EMR

Depois de criar seu aplicativo e configurar as permissões necessárias, use as etapas a seguir para executar um notebook interativo com o EMR Studio:

1. Navegue até a guia Espaços de trabalho no EMR Studio. Se você ainda precisar configurar um local de armazenamento do Amazon S3 e uma função de serviço do EMR Studio, selecione o botão Configurar estúdio no banner na parte superior da tela.
2. Para acessar um notebook, selecione um espaço de trabalho ou crie um novo espaço de trabalho. Use o Início rápido para abrir seu espaço de trabalho em uma nova guia.
3. Vá para a guia recém-aberta. Selecione o ícone Computar no painel de navegação à esquerda. Selecione EMR Sem servidor como o tipo de computação.
4. Selecione o aplicativo interativo ativado que você criou na seção anterior.
5. No campo Função de tempo de execução, insira o nome da IAM função que seu aplicativo EMR sem servidor pode assumir para a execução do trabalho. Para saber mais sobre as funções de tempo de execução, consulte Job runtime roles no Amazon EMR Serverless User Guide.
6. Selecione Anexar. Isso pode levar até um minuto. A página será atualizada quando anexada.
7. Escolha um kernel e inicie um notebook. Você também pode procurar exemplos de cadernos no EMR Serverless e copiá-los para o seu espaço de trabalho. Para acessar os cadernos de exemplo, navegue até o **`{...}`**menu na navegação à esquerda e navegue pelos cadernos que têm `serverless` o nome do arquivo do notebook.
8. No notebook, você pode acessar o link do registro do driver e um link para a interface do usuário do Apache Spark, uma interface em tempo real que fornece métricas para monitorar seu trabalho. Para obter mais informações, consulte Monitoramento de aplicativos e trabalhos EMR sem servidor no Guia do usuário do Amazon EMR Serverless.

Quando você anexa um aplicativo a um espaço de trabalho do Studio, o início do aplicativo é acionado automaticamente se ainda não estiver em execução. Você também pode pré-iniciar o aplicativo e mantê-lo pronto antes de anexá-lo ao espaço de trabalho.

Etapa 3: Exibir a interface do usuário e os registros do aplicativo

Para visualizar a interface do usuário do aplicativo, primeiro identifique a execução do trabalho. Uma opção para a interface do Spark ou a interface do usuário do Hive Tez está disponível na primeira linha de opções para a execução desse trabalho, com base no tipo de trabalho. Selecione a opção apropriada.

Se você escolheu a interface do Spark, escolha a guia Executors para ver os registros do driver e dos executores. Se você escolheu a interface do usuário do Hive Tez, escolha a guia Todas as tarefas para visualizar os registros.

Depois que o status de execução do trabalho for exibido como Sucesso, você poderá visualizar a saída do trabalho em seu bucket do S3.

Etapa 4: limpar

Embora o aplicativo que você criou deva parar automaticamente após 15 minutos de inatividade, ainda recomendamos que você libere recursos que não pretende usar novamente.

Para excluir o aplicativo, navegue até a página Listar aplicativos. Selecione o aplicativo que você criou e escolha Ações → Parar para interromper o aplicativo. Depois que o aplicativo estiver no `STOPPED` estado, selecione o mesmo aplicativo e escolha Ações → Excluir.

Para obter mais exemplos de execução de trabalhos do Spark e do Hive, consulte e. Trabalhos do Spark Trabalhos na Hive

Começando a partir do AWS CLI

Etapa 1: criar um aplicativo EMR sem servidor

Use o `emr-serverless create-application`comando para criar seu primeiro aplicativo EMR sem servidor. Você precisa especificar o tipo de aplicativo e a etiqueta de EMR lançamento da Amazon associada à versão do aplicativo que você deseja usar. O nome do aplicativo é opcional.

Spark

Para criar um aplicativo Spark, execute o comando a seguir.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
```

```
    --type "SPARK" \
    --name my-application
```

Hive

Para criar um aplicativo Hive, execute o comando a seguir.

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

Observe o ID do aplicativo retornado na saída. Você usará o ID para iniciar a inscrição e durante o envio do trabalho, depois chamado de*`application-id`*.

Antes de prosseguir paraEtapa 2: enviar uma execução de trabalho para seu aplicativo sem EMR servidor, certifique-se de que seu aplicativo tenha atingido o `CREATED` estado com `get-application`APIo.

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

EMRO Serverless cria trabalhadores para acomodar os trabalhos solicitados. Por padrão, eles são criados sob demanda, mas você também pode especificar uma capacidade pré-inicializada definindo o `initialCapacity` parâmetro ao criar o aplicativo. Você também pode limitar a capacidade máxima total que um aplicativo pode usar com o `maximumCapacity` parâmetro. Para saber mais sobre essas opções, consulte Configurando um aplicativo.

Etapa 2: enviar uma execução de trabalho para seu aplicativo sem EMR servidor

Agora, seu aplicativo EMR sem servidor está pronto para executar trabalhos.

Spark

Nesta etapa, usamos um PySpark script para calcular o número de ocorrências de palavras exclusivas em vários arquivos de texto. Um bucket S3 público e somente para leitura armazena o script e o conjunto de dados. O aplicativo envia o arquivo de saída e os dados de log do tempo de execução do Spark `/output` e os `/logs` diretórios no bucket do S3 que você criou.

Para executar uma tarefa do Spark

1. Use o comando a seguir para copiar o script de amostra que executaremos em seu novo bucket.

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
```

2. No comando a seguir, *application-id* substitua pelo ID do seu aplicativo. *job-role-arn*Substitua pela função de tempo de execução em ARN que você criouCrie uma função de tempo de execução do trabalho. Substituto *job-run-name* com o nome que você deseja chamar de execução do trabalho. Substitua todas *DOC-EXAMPLE-BUCKET* as strings pelo bucket do Amazon S3 que você criou e `/output` adicione ao caminho. Isso cria uma nova pasta em seu bucket na qual o EMR Serverless pode copiar os arquivos de saída do seu aplicativo.

```
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://DOC-EXAMPLE-BUCKET/scripts/wordcount.py",
          "entryPointArguments": ["s3://DOC-EXAMPLE-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. Observe o ID de execução do trabalho retornado na saída. *job-run-id*Substitua por essa ID nas etapas a seguir.

Hive

Neste tutorial, criamos uma tabela, inserimos alguns registros e executamos uma consulta de agregação de contagem. Para executar o trabalho do Hive, primeiro crie um arquivo que contenha todas as consultas do Hive a serem executadas como parte de um único trabalho, carregue o arquivo no S3 e especifique esse caminho do S3 ao iniciar o trabalho do Hive.

Para executar um trabalho do Hive

1. Crie um arquivo chamado `hive-query.ql` que contenha todas as consultas que você deseja executar em seu trabalho do 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. Faça `hive-query.ql` o upload para seu bucket do S3 com o comando a seguir.

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-
query.ql
```

3. No comando a seguir, *`application-id`* substitua pelo seu próprio ID do aplicativo. *`job-role-arn`*Substitua pela função de tempo de execução em ARN que você criouCrie uma função de tempo de execução do trabalho. Substitua todas *`DOC-EXAMPLE-BUCKET`* as cadeias de caracteres pelo bucket do Amazon S3 que você criou `/logs` e `/output` adicione-a ao caminho. Isso cria novas pastas em seu bucket, nas quais o EMR Serverless pode copiar os arquivos de saída e de log do seu aplicativo.

```
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"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs"
           }
        }
    }'
```

4. Observe o ID de execução do trabalho retornado na saída. *job-run-id*Substitua por essa ID nas etapas a seguir.

Etapa 3: Revise a saída da execução do seu trabalho

Normalmente, a execução do trabalho deve levar de 3 a 5 minutos para ser concluída.

Spark

Você pode verificar o estado do seu trabalho do Spark com o comando a seguir.

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

Com o destino do registro definido como`s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/logs`, você pode encontrar os registros desse trabalho específico executado em`s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/logs/applications/application-id/jobs/job-run-id`.

Para aplicativos Spark, o EMR Serverless envia registros de eventos a cada 30 segundos para a `sparklogs` pasta no destino do registro do S3. Quando seu trabalho é concluído, os registros de tempo de execução do Spark para o driver e os executores são enviados para pastas nomeadas apropriadamente pelo tipo de trabalhador, como ou. `driver` `executor` A saída do PySpark trabalho é enviada para o. `s3://DOC-EXAMPLE-BUCKET/output/`

Hive

Você pode verificar o estado do seu trabalho do Hive com o comando a seguir.

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

Com o destino do registro definido como`s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs`, você pode encontrar os registros desse trabalho específico executado em`s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id`.

Para aplicativos Hive, o EMR Serverless carrega continuamente o driver do Hive para a pasta e os registros de tarefas do Tez para a `HIVE_DRIVER` pasta do seu destino de log do `TEZ_TASK` S3. Depois que a execução do trabalho atinge o `SUCCEEDED` estado, a saída da sua consulta do Hive fica disponível no local do Amazon S3 que você especificou `monitoringConfiguration` no campo de. `configurationOverrides`

Etapa 4: limpar

Ao terminar de trabalhar com este tutorial, considere excluir os recursos que você criou. Recomendamos que você libere recursos que não pretende usar novamente.

Exclua seu aplicativo

Para excluir um aplicativo, use o comando a seguir.

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

Exclua seu bucket de log do S3

Para excluir seu bucket de registro e saída do S3, use o comando a seguir. *DOC-EXAMPLE-BUCKET*Substitua pelo nome real do bucket S3 criado em.. Prepare o armazenamento sem EMR servidor

```
aws s3 rm s3://DOC-EXAMPLE-BUCKET --recursive
aws s3api delete-bucket --bucket DOC-EXAMPLE-BUCKET
```

Exclua sua função de tempo de execução do trabalho

Para excluir a função de tempo de execução, separe a política da função. Em seguida, você pode excluir a função e a política.

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

Para excluir a função, use o comando a seguir.

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

Para excluir a política que foi anexada à função, use o comando a seguir.

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

Para obter mais exemplos de execução de trabalhos do Spark e do Hive, consulte e. Trabalhos do Spark Trabalhos na Hive

Interagindo com um aplicativo

Esta seção aborda como você pode interagir com seu aplicativo Amazon EMR Serverless com o AWS CLI e os padrões para os motores Spark e Hive.

Tópicos

Estados do aplicativo

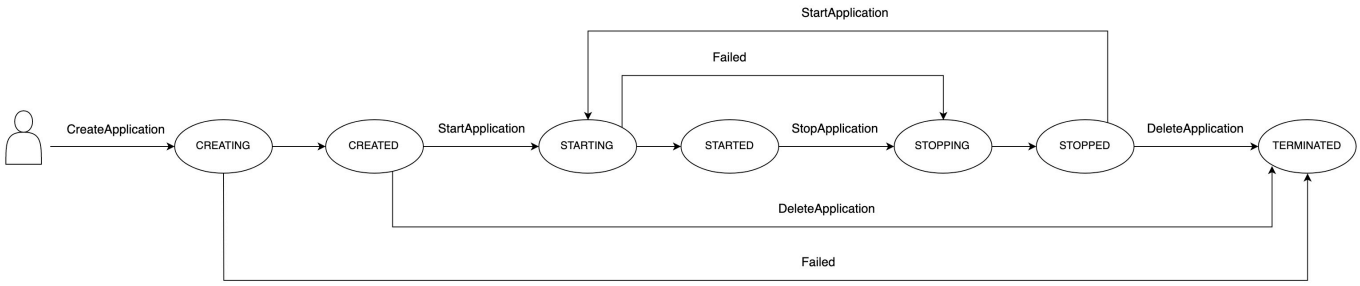
Quando você cria um aplicativo com EMR Serverless, a execução do aplicativo entra no `CREATING` estado. Em seguida, ela passa pelos estados apresentados a seguir até obter êxito (sair com o código `0`) ou falhar (sair com um código diferente de zero).

Os aplicativos podem ter os seguintes estados:

Estado	Descrição
Criando	O aplicativo está sendo preparado e ainda não está pronto para uso.
Criado	O aplicativo foi criado, mas ainda não provision ou a capacidade. Você pode modificar o aplicativo para alterar sua configuração de capacidade inicial.
Starting	O aplicativo está sendo iniciado e está provisionando a capacidade.

Estado	Descrição
Iniciado	O aplicativo está pronto para aceitar novos empregos. O aplicativo só aceita vagas quando está nesse estado.
Stopping	Todos os trabalhos foram concluídos e o aplicativo está liberando sua capacidade.
Interrompido	O aplicativo está parado e nenhum recurso está sendo executado no aplicativo. Você pode modificar o aplicativo para alterar sua configura ção de capacidade inicial.
Encerrado	O aplicativo foi encerrado e não aparece na sua lista de aplicativos.

O diagrama a seguir mostra a trajetória dos estados dos aplicativos EMR sem servidor.



Interagindo com seu aplicativo a partir do console do EMR Studio

No console do EMR Studio, você pode criar, visualizar e gerenciar aplicativos EMR sem servidor. Para navegar até o console do EMR Studio, siga as instruções em Introdução ao console.

Cria uma aplicação

Com a página Criar aplicativo, você pode criar um aplicativo EMR sem servidor seguindo estas etapas.

1. No campo Nome, insira o nome que você deseja chamar de seu aplicativo.
2. No campo Tipo, escolha Spark ou Hive como o tipo do aplicativo.
3. No campo Versão da versão, escolha o número da EMR versão.
4. Nas opções de Arquitetura, escolha a arquitetura do conjunto de instruções a ser usada. Para obter mais informações, consulte Opções de EMR arquitetura Amazon Serverless.

   - arm64 — ARM arquitetura de 64 bits; para usar processadores Graviton
   - x86_64 — arquitetura x86 de 64 bits; para usar processadores baseados em x86

5. Há duas opções de configuração do aplicativo para os campos restantes: configurações padrão e configurações personalizadas. Esses campos são opcionais.

   Configurações padrão — As configurações padrão permitem que você crie rapidamente um aplicativo com capacidade pré-inicializada. Isso inclui um driver e um executor para o Spark e um driver e um Tez Task para o Hive. As configurações padrão não permitem a conectividade de rede com o seuVPCs. O aplicativo está configurado para parar se ficar ocioso por 15 minutos e iniciar automaticamente no envio do trabalho.

   Configurações personalizadas — As configurações personalizadas permitem que você modifique as propriedades a seguir.

   - Capacidade pré-inicializada — O número de motoristas e executores dos trabalhadores do Hive Tez Task e o tamanho de cada trabalhador.
   - Limites do aplicativo — A capacidade máxima de um aplicativo.
   - Comportamento do aplicativo — O comportamento de inicialização e parada automáticas do aplicativo.
   - Conexões de rede — Conectividade de rede com VPC recursos.
   - Tags — Tags personalizadas que você pode atribuir ao aplicativo.

   Para obter mais informações sobre capacidade pré-inicializada, limites de aplicativos e comportamento de aplicativos, consulte. Configurando um aplicativo Para obter mais informações sobre conectividade de rede, consulteConfigurando o acesso VPC.

6. Para criar o aplicativo, escolha Criar aplicativo.

Listar aplicativos

Você pode visualizar todos os aplicativos EMR sem servidor existentes na página Listar aplicativos. Você pode escolher o nome de um aplicativo para navegar até a página Detalhes desse aplicativo.

Gerenciar aplicações

Você pode realizar as seguintes ações em um aplicativo na página Listar aplicativos ou na página Detalhes de um aplicativo específico.

Iniciar aplicativo

Escolha essa opção para iniciar manualmente um aplicativo.

Interromper o aplicativo

Escolha essa opção para interromper manualmente um aplicativo. Um aplicativo não deve ter trabalhos em execução para ser interrompido. Para saber mais sobre as transições de estado do aplicativo, consulteEstados do aplicativo.

Configurar aplicativo

Edite as configurações opcionais de um aplicativo na página Configurar aplicativo. Você pode alterar a maioria das configurações do aplicativo. Por exemplo, você pode alterar o rótulo de lançamento de um aplicativo para atualizá-lo para uma versão diferente da Amazon EMR ou pode mudar a arquitetura de x86_64 para arm64. As outras configurações opcionais são as mesmas que estão na seção Configurações personalizadas na página Criar aplicativo. Para obter mais informações sobre as configurações do aplicativo, consulteCria uma aplicação.

Excluir aplicativo

Escolha essa opção para excluir manualmente um aplicativo. Você deve interromper um aplicativo para excluí-lo. Para saber mais sobre as transições de estado do aplicativo, consulteEstados do aplicativo.

Interagindo com seu aplicativo no AWS CLI

Do AWS CLI, você pode criar, descrever e excluir aplicativos individuais. Você também pode listar todos os seus aplicativos para poder visualizá-los rapidamente. Esta seção descreve como realizar essas ações. Para mais operações de aplicativos, como iniciar, interromper e atualizar seu aplicativo,

consulte a Referência EMRsem servidor API. Para exemplos de como usar o EMR Serverless API usando o AWS SDK for Java, veja exemplos de Java em nosso GitHub repositório. Para exemplos de como usar o EMR Serverless API usando o AWS SDK for Python (Boto), veja exemplos de Python em nosso GitHub repositório.

Para criar um aplicativo, use`create-application`. Você deve especificar `SPARK` ou `HIVE` como aplicativo`type`. Esse comando retorna o nomeARN, o nome e o ID do aplicativo.

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

Para descrever um aplicativo, use `get-application` e forneça seu`application-id`. Esse comando retorna as configurações relacionadas ao estado e à capacidade do seu aplicativo.

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

Para listar todos os seus aplicativos, ligue`list-applications`. Esse comando retorna as mesmas propriedades, `get-application` mas inclui todos os seus aplicativos.

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

Para excluir seu aplicativo, ligue `delete-application` e forneça seu`application-id`.

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

Configurando um aplicativo

Com o EMR Serverless, você pode configurar os aplicativos que você usa. Por exemplo, você pode definir a capacidade máxima para a qual um aplicativo pode ser expandido, configurar a capacidade pré-inicializada para manter o motorista e os trabalhadores prontos para responder e especificar um conjunto comum de configurações de tempo de execução e monitoramento no nível do aplicativo. As páginas a seguir descrevem como configurar aplicativos ao usar o EMR Serverless.

Tópicos

Entendendo o comportamento do aplicativo

Comportamento padrão do aplicativo

Início automático — Por padrão, um aplicativo é configurado para iniciar automaticamente no envio do trabalho. Você pode desativar esse recurso.

Parada automática — Por padrão, um aplicativo é configurado para parar automaticamente quando ocioso por 15 minutos. Quando um aplicativo muda para o `STOPPED` estado, ele libera qualquer capacidade pré-inicializada configurada. Você pode modificar a quantidade de tempo ocioso antes que um aplicativo pare automaticamente ou pode desativar esse recurso.

Capacidade máxima

Você pode configurar a capacidade máxima para a qual um aplicativo pode ser escalado. Você pode especificar sua capacidade máxima em termos CPU de memória (GB) e disco (GB).

> **Note**
>
> Recomendamos configurar sua capacidade máxima para ser proporcional aos tamanhos de trabalhadores suportados, multiplicando o número de trabalhadores por seus tamanhos. Por exemplo, se você quiser limitar seu aplicativo a 50 trabalhadores com 2vCPUs, 16 GB para memória e 20 GB para disco, defina sua capacidade máxima para 100vCPUs, 800 GB para memória e 1000 GB para disco.

Configurações de trabalhadores suportadas

A tabela a seguir mostra as configurações e tamanhos de trabalhadores compatíveis que você pode especificar para o EMR Serverless. Você pode configurar tamanhos diferentes para drivers e executores com base na necessidade de sua carga de trabalho.

CPU	Memória	Armazenamento efêmero padrão
1 v CPU	Mínimo de 2 GB, máximo de 8 GB, em incrementos de 1 GB	20 GB - 200 GB
2 v CPU	Mínimo de 4 GB, máximo de 16 GB, em incrementos de 1 GB	20 GB - 200 GB
4 v CPU	Mínimo de 8 GB, máximo de 30 GB, em incrementos de 1 GB	20 GB - 200 GB
8 v CPU	Mínimo de 16 GB, máximo de 60 GB, em incrementos de 4 GB	20 GB - 200 GB
16 g CPU	Mínimo de 32 GB, máximo de 120 GB, em incrementos de 8 GB	20 GB - 200 GB

CPU— Cada trabalhador pode ter 1, 2, 4, 8 ou 16vCPUs.

Memória — Cada trabalhador tem memória, especificada em GB, dentro dos limites listados na tabela anterior. Os trabalhos do Spark têm uma sobrecarga de memória, o que significa que a memória que eles usam é maior do que os tamanhos de contêineres especificados. Essa sobrecarga é especificada com as propriedades `spark.driver.memory0verhead` e. `spark.executor.memory0verhead` A sobrecarga tem um valor padrão de 10% da memória do contêiner, com um mínimo de 384 MB. Você deve considerar essa sobrecarga ao escolher o tamanho dos trabalhadores.

Por exemplo, se você escolher 4 vCPUs para sua instância de trabalho e uma capacidade de armazenamento pré-inicializada de 30 GB, defina um valor de aproximadamente 27 GB como memória executora para sua tarefa do Spark. Isso maximiza a utilização de sua capacidade pré-inicializada. A memória utilizável seria de 27 GB, mais 10% de 27 GB (2,7 GB), totalizando 29,7 GB.

Disco — Você pode configurar cada trabalhador com discos de armazenamento temporário com tamanho mínimo de 20 GB e máximo de 200 GB. Você paga apenas pelo armazenamento adicional além de 20 GB configurado por trabalhador.

Capacidade pré-inicializada

EMRO Serverless fornece um recurso opcional que mantém o motorista e os trabalhadores pré-inicializados e prontos para responder em segundos. Isso cria efetivamente um grupo caloroso de trabalhadores para um aplicativo. Esse recurso é chamado de capacidade pré-inicializada. Para configurar esse recurso, você pode definir o `initialCapacity` parâmetro de um aplicativo para o número de trabalhadores que você deseja pré-inicializar. Com a capacidade de trabalho pré-inicializada, os trabalhos começam imediatamente. Isso é ideal quando você deseja implementar aplicativos iterativos e trabalhos urgentes.

Quando você envia um trabalho, se houver trabalhadores disponíveis, o trabalho usa esses recursos para iniciar sua execução. `initialCapacity` Se esses trabalhadores já estiverem sendo usados por outros trabalhos, ou se o trabalho precisar de mais recursos do que os disponíveis`initialCapacity`, o aplicativo solicitará e obterá trabalhadores adicionais, até os limites máximos de recursos definidos para o aplicativo. Quando um trabalho termina sua execução, ele libera os trabalhadores que usou e o número de recursos disponíveis para o aplicativo retorna para`initialCapacity`. Um aplicativo mantém os `initialCapacity` recursos mesmo após o término da execução dos trabalhos. O aplicativo libera recursos em excesso depois de `initialCapacity` os trabalhos não precisarem mais deles para serem executados.

A capacidade pré-inicializada está disponível e pronta para uso quando o aplicativo é iniciado. A capacidade pré-inicializada fica inativa quando o aplicativo é interrompido. Um aplicativo passa para o `STARTED` estado somente se a capacidade pré-inicializada solicitada tiver sido criada e estiver pronta para uso. Durante todo o tempo em que o aplicativo está no `STARTED` estado, o EMR Serverless mantém a capacidade pré-inicializada disponível para uso ou em uso por trabalhos ou cargas de trabalho interativas. O recurso restaura a capacidade de contêineres liberados ou com defeito. Isso mantém o número de trabalhadores que o `InitialCapacity` parâmetro especifica. O estado de um aplicativo sem capacidade pré-inicializada pode mudar imediatamente de `CREATED` para. `STARTED`

Você pode configurar o aplicativo para liberar a capacidade pré-inicializada se ela não for usada por um determinado período de tempo, com um padrão de 15 minutos. Uma inscrição interrompida é iniciada automaticamente quando você envia um novo trabalho. Você pode definir essas

configurações automáticas de início e parada ao criar o aplicativo ou alterá-las quando o aplicativo estiver em um `STOPPED` estado `CREATED` ou.

Você pode alterar as `InitialCapacity` contagens e especificar configurações computacionaisCPU, como memória e disco, para cada trabalhador. Como você não pode fazer modificações parciais, você deve especificar todas as configurações de computação ao alterar os valores. Você só pode alterar as configurações quando o aplicativo está no `STOPPED` estado `CREATED` ou.

> **Note**
>
> Para otimizar o uso dos recursos do seu aplicativo, recomendamos alinhar os tamanhos dos contêineres com os tamanhos pré-inicializados dos funcionários de capacidade. Por exemplo, se você configurar o tamanho do executor do Spark para 2 CPUs e sua memória para 8 GB, mas o tamanho do operador de capacidade pré-inicializada for 4 CPUs com 16 GB de memória, os executores do Spark usarão apenas metade dos recursos dos trabalhadores quando forem designados para esse trabalho.

Personalizando a capacidade pré-inicializada para Spark e Hive

Você pode personalizar ainda mais a capacidade pré-inicializada para cargas de trabalho executadas em estruturas específicas de big data. Por exemplo, quando uma carga de trabalho é executada no Apache Spark, você pode especificar quantos trabalhadores começam como drivers e quantos começam como executores. Da mesma forma, ao usar o Apache Hive, você pode especificar quantos trabalhadores começam como drivers do Hive e quantos devem executar tarefas do Tez.

Configurando um aplicativo executando o Apache Hive com capacidade pré-inicializada

A API solicitação a seguir cria um aplicativo executando o Apache Hive com base na EMR versão emr-6.6.0 da Amazon. O aplicativo começa com 5 drivers Hive pré-inicializados, cada um com 2 v CPU e 4 GB de memória, e 50 trabalhadores de tarefas Tez pré-inicializados, cada um com 4 v CPU e 8 GB de memória. Quando as consultas do Hive são executadas nesse aplicativo, elas primeiro usam os trabalhadores pré-inicializados e começam a ser executadas imediatamente. Se todos os trabalhadores pré-inicializados estiverem ocupados e mais trabalhos do Hive forem enviados, o aplicativo poderá ser expandido para um total de 400 v CPU e 1024 GB de memória. Opcionalmente, você pode omitir a capacidade do funcionário `DRIVER` ou do `TEZ_TASK` trabalhador.

```
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"
  }'
```

Configurando um aplicativo executando o Apache Spark com capacidade pré-inicializada

A API solicitação a seguir cria um aplicativo que executa o Apache Spark 3.2.0 com base na versão 6.6.0 da AmazonEMR. O aplicativo começa com 5 drivers Spark pré-inicializados, cada um com 2 v CPU e 4 GB de memória, e 50 executores pré-inicializados, cada um com 4 v CPU e 8 GB de memória. Quando as tarefas do Spark são executadas nesse aplicativo, elas primeiro usam os trabalhadores pré-inicializados e começam a ser executadas imediatamente. Se todos os trabalhadores pré-inicializados estiverem ocupados e mais trabalhos do Spark forem enviados, o aplicativo poderá ser expandido para um total de 400 v CPU e 1024 GB de memória. Opcionalmente, você pode omitir a capacidade do `DRIVER` ou do. `EXECUTOR`

> **Note**
>
> O Spark adiciona uma sobrecarga de memória configurável, com um valor padrão de 10%, à memória solicitada pelo driver e pelos executores. Para que os trabalhos usem trabalhadores

pré-inicializados, a configuração inicial da memória de capacidade deve ser maior do que a memória solicitada pelo trabalho e pela sobrecarga.

```
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"
  }'
```

Configuração padrão do aplicativo para EMR Serverless

Você pode especificar um conjunto comum de configurações de tempo de execução e monitoramento no nível do aplicativo para todos os trabalhos que você envia no mesmo aplicativo. Isso reduz a sobrecarga adicional associada à necessidade de enviar as mesmas configurações para cada trabalho.

Você pode modificar as configurações nos seguintes momentos:

- Declare as configurações em nível de aplicativo no envio do trabalho.
- Substitua as configurações padrão durante a execução do trabalho.

As seções a seguir fornecem mais detalhes e um exemplo para contextualizar melhor.

Declarando configurações no nível do aplicativo

Você pode especificar as propriedades de registro em nível de aplicativo e configuração de tempo de execução para os trabalhos que você envia no aplicativo.

`monitoringConfiguration`

Para especificar as configurações de log para trabalhos que você envia com o aplicativo, use o `monitoringConfiguration`campo. Para obter mais informações sobre o registro no EMR Serverless, consulte. Armazenando registros

`runtimeConfiguration`

Para especificar propriedades de configuração de tempo de execução`spark-defaults`, como, forneça um objeto de configuração no `runtimeConfiguration` campo. Isso afeta as configurações padrão de todos os trabalhos que você envia com o aplicativo. Para ter mais informações, consulte Parâmetro de substituição da configuração do Hive e Parâmetro de substituição da configuração do Spark.

As classificações de configuração disponíveis variam de acordo com a versão específica do EMR Serverless. Por exemplo, classificações para Log4j personalizado `spark-executor-log4j2` estão disponíveis somente nas versões 6.8.0 `spark-driver-log4j2` e superiores. Para obter uma lista de propriedades específicas do aplicativo, consulte e. Propriedades de trabalho no Spark Propriedades de trabalho em Hive

Você também pode configurar as propriedades do Apache Log4j2, AWS Secrets Manager para proteção de dados e tempo de execução do Java 17 no nível do aplicativo.

Para transmitir segredos do Secrets Manager no nível do aplicativo, anexe a política a seguir aos usuários e funções que precisam criar ou atualizar aplicativos EMR sem servidor com segredos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerPolicy",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
```

```
                "secretsmanager:DescribeSecret",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:secretsmanager:your-secret-arn"
        }
    ]
}
```

Para obter mais informações sobre a criação de políticas personalizadas para segredos, consulte Exemplos de políticas de permissões para AWS Secrets Manager no AWS Secrets Manager Guia do usuário.

> **Note**
>
> O `runtimeConfiguration` que você especifica no nível do aplicativo é `applicationConfiguration` mapeado para no `StartJobRun`API.

Exemplo de declaração

O exemplo a seguir mostra como declarar configurações padrão com. `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://DOC-EXAMPLE-BUCKET-LOGGING/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

Substituindo configurações durante a execução de um trabalho

Você pode especificar substituições de configuração para a configuração do aplicativo e a configuração de monitoramento com o. `StartJobRun`API EMRO Serverless então mescla as configurações que você especifica no nível do aplicativo e no nível do trabalho para determinar as configurações para a execução do trabalho.

O nível de granularidade quando a mesclagem ocorre é o seguinte:

- **`ApplicationConfiguration`**- Tipo de classificação, por exemplo`spark-defaults`.
- **`MonitoringConfiguration`**- Tipo de configuração, por exemplo`s3MonitoringConfiguration`.

> **Note**
>
> A prioridade das configurações que você fornece `StartJobRun`substitui as configurações que você fornece no nível do aplicativo.

Para obter mais informações sobre classificações prioritárias, consulte Parâmetro de substituição da configuração do Hive e. Parâmetro de substituição da configuração do Spark

Quando você inicia um trabalho, se você não especificar uma configuração específica, ela será herdada do aplicativo. Se você declarar as configurações no nível do trabalho, poderá realizar as seguintes operações:

- Substituir uma configuração existente - Forneça o mesmo parâmetro de configuração na `StartJobRun` solicitação com seus valores de substituição.
- Adicionar uma configuração adicional - Adicione o novo parâmetro de configuração na `StartJobRun` solicitação com os valores que você deseja especificar.
- Remover uma configuração existente - Para remover uma configuração de tempo de execução do aplicativo, forneça a chave para a configuração que você deseja remover e passe uma declaração vazia `{}` para a configuração. Não recomendamos remover nenhuma classificação que contenha parâmetros necessários para a execução de um trabalho. Por exemplo, se você tentar remover as propriedades necessárias para uma tarefa do Hive, a tarefa falhará.

  Para remover uma configuração de monitoramento de aplicativos, use o método apropriado para o tipo de configuração relevante:

  - **`cloudWatchLoggingConfiguration`**- Para remover`cloudWatchLogging`, passe a bandeira habilitada como`false`.
  - **`managedPersistenceMonitoringConfiguration`**- Para remover as configurações de persistência gerenciada e voltar ao estado habilitado padrão, passe uma declaração vazia `{}` para a configuração.
  - **`s3MonitoringConfiguration`**- Para remover`s3MonitoringConfiguration`, passe uma declaração vazia `{}` para a configuração.

Exemplo de substituição

O exemplo a seguir mostra diferentes operações que você pode realizar durante o envio do trabalho em`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://DOC-EXAMPLE-BUCKET-OUTPUT/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
            }
        }
    }'
```

No momento da execução do trabalho, as seguintes classificações e configurações serão aplicadas com base na classificação de substituição de prioridade descrita em e. Parâmetro de substituição da configuração do Hive Parâmetro de substituição da configuração do Spark

- A classificação `spark-defaults` será atualizada com as propriedades especificadas no nível do cargo. Somente as propriedades incluídas em `StartJobRun` seriam consideradas para essa classificação.
- A classificação `spark-executor-log4j2` será adicionada à lista existente de classificações.
- A classificação `spark-driver-log4j2` será removida.
- As configurações do `managedPersistenceMonitoringConfiguration` serão atualizadas com as configurações no nível do trabalho.
- As configurações do `s3MonitoringConfiguration` serão removidas.
- As configurações do `cloudWatchLoggingConfiguration` serão adicionadas às configurações de monitoramento existentes.

Personalizando uma imagem sem EMR servidor

A partir do Amazon EMR 6.9.0, você pode usar imagens personalizadas para empacotar dependências de aplicativos e ambientes de execução em um único contêiner com o Amazon Serverless. EMR Isso simplifica a forma como você gerencia as dependências da carga de trabalho e torna seus pacotes mais portáteis. Quando você personaliza sua imagem EMR sem servidor, ela oferece os seguintes benefícios:

- Instala e configura pacotes otimizados para suas cargas de trabalho. Esses pacotes podem não estar amplamente disponíveis na distribuição pública dos ambientes de EMR execução da Amazon.
- Integra o EMR Serverless aos processos atuais de criação, teste e implantação estabelecidos em sua organização, incluindo desenvolvimento e teste locais.
- Aplica processos de segurança estabelecidos, como digitalização de imagens, que atendem aos requisitos de conformidade e governança em sua organização.
- Permite que você use suas próprias versões do JDK Python para seus aplicativos.

EMRO Serverless fornece imagens que você pode usar como base ao criar suas próprias imagens. A imagem base fornece os jars, a configuração e as bibliotecas essenciais para a imagem interagir com o EMR Serverless. Você pode encontrar a imagem base na Amazon ECR Public Gallery. Use a imagem que corresponda ao seu tipo de aplicativo (Spark ou Hive) e à versão de lançamento. Por exemplo, se você criar um aplicativo na EMR versão 6.9.0 da Amazon, use as imagens a seguir.

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

Pré-requisitos

Antes de criar uma imagem personalizada EMR sem servidor, preencha esses pré-requisitos.

1. Crie um ECR repositório da Amazon no mesmo Região da AWS que você usa para iniciar aplicativos EMR sem servidor. Para criar um repositório ECR privado da Amazon, consulte Criação de um repositório privado.
2. Para conceder aos usuários acesso ao seu ECR repositório da Amazon, adicione as seguintes políticas aos usuários e funções que criam ou atualizam aplicativos EMR sem servidor com imagens desse repositório.

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

```
}
```

Para obter mais exemplos de políticas ECR baseadas em identidade da Amazon, consulte exemplos de políticas baseadas em identidade do Amazon Elastic Container Registry.

Etapa 1: criar uma imagem personalizada a partir de imagens base EMR sem servidor

Primeiro, crie um Dockerfile que comece com uma `FROM` instrução que usa sua imagem base preferida. Após a `FROM` instrução, você pode incluir qualquer modificação que desejar fazer na imagem. A imagem base define automaticamente o valor `USER` para`hadoop`. Essa configuração pode não ter permissões para todas as modificações que você inclui. Como solução alternativa, defina o `USER` para`root`, modifique sua imagem e, em seguida, defina o de `USER` volta para`hadoop:hadoop`. Para ver exemplos de casos de uso comuns, consulteUsando imagens personalizadas com o EMR Serverless.

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

USER root
# MODIFICATIONS GO HERE

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Depois de ter o Dockerfile, crie a imagem com o comando a seguir.

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

Etapa 2: validar a imagem localmente

EMRO Serverless fornece uma ferramenta off-line que pode verificar estaticamente sua imagem personalizada para validar arquivos básicos, variáveis de ambiente e configurações corretas de imagem. Para obter informações sobre como instalar e executar a ferramenta, consulte a Amazon EMR Serverless Image. CLI GitHub

Depois de instalar a ferramenta, execute o comando a seguir para validar uma imagem:

```
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
```

Você deve ver uma saída semelhante à seguinte.

```
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.
----------------------------------------------------------------
```

Etapa 3: Faça o upload da imagem para o seu ECR repositório da Amazon

Envie sua ECR imagem da Amazon para o ECR repositório da Amazon com os seguintes comandos. Verifique se você tem as IAM permissões corretas para enviar a imagem para o seu repositório. Para obter mais informações, consulte Como enviar uma imagem no Guia do ECR usuário da Amazon.

```
# 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
```

Etapa 4: criar ou atualizar um aplicativo com imagens personalizadas

Selecione o AWS Management Console aba ou AWS CLI separe a guia de acordo com a forma como você deseja iniciar seu aplicativo e, em seguida, conclua as etapas a seguir.

Console

1. Faça login no console do EMR Studio em https://console.aws.amazon.com/emr. Navegue até seu aplicativo ou crie um novo aplicativo com as instruções em Criar um aplicativo.
2. Para especificar imagens personalizadas ao criar ou atualizar um aplicativo EMR sem servidor, selecione Configurações personalizadas nas opções de configuração do aplicativo.
3. Na seção Configurações de imagem personalizada, marque a caixa de seleção Usar a imagem personalizada com este aplicativo.
4. Cole a ECR imagem da Amazon URI no URI campo Imagem. EMRO Serverless usa essa imagem para todos os tipos de trabalhadores do aplicativo. Como alternativa, você pode escolher imagens personalizadas diferentes e colar ECR imagens diferentes da Amazon URIs para cada tipo de trabalhador.

CLI

- Crie um aplicativo com o `image-configuration` parâmetro. EMRO Serverless aplica essa configuração a todos os tipos de trabalhadores.

  ```
  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"
  }'
  ```

  Para criar um aplicativo com configurações de imagem diferentes para cada tipo de trabalhador, use o `worker-type-specifications` parâmetro.

```
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"
        }
    }
}'
```

Para atualizar um aplicativo, use o `image-configuration` parâmetro. EMRO Serverless aplica essa configuração a todos os tipos de trabalhadores.

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

Etapa 5: permitir que o EMR Serverless acesse o repositório de imagens personalizado

Adicione a seguinte política de recursos ao ECR repositório da Amazon para permitir que o responsável pelo serviço EMR Serverless use as `download` solicitaçõesget,`describe`, e desse repositório.

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

```
      "Sid": "Emr Serverless Custom Image Support",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition":{
        "StringEquals":{
          "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
        }
      }
    }
  ]
}
```

Como prática recomendada de segurança, adicione uma chave de `aws:SourceArn` condição à política do repositório. A chave de condição IAM global `aws:SourceArn` garante que o EMR Serverless use o repositório somente para um aplicativo. ARN Para obter mais informações sobre as políticas de ECR repositórios da Amazon, consulte Criação de um repositório privado.

Considerações e limitações

Ao trabalhar com imagens personalizadas, considere o seguinte:

- Use a imagem base correta que corresponda ao tipo (Spark ou Hive) e à etiqueta de lançamento (por exemplo,`emr-6.9.0`) do seu aplicativo.
- EMRO Serverless `[CMD]` ignora nossas `[ENTRYPOINT]` instruções no arquivo Docker. Use instruções comuns no arquivo Docker, como `[COPY][RUN]`, e. `[WORKDIR]`
- Você não deve modificar as variáveis de ambiente `JAVA_HOMESPARK_HOME`,`HIVE_HOME`,, `TEZ_HOME` ao criar uma imagem personalizada.
- As imagens personalizadas não podem exceder 5 GB de tamanho.
- Se você modificar binários ou jars nas imagens EMR básicas da Amazon, isso poderá causar falhas no lançamento de aplicativos ou trabalhos.
- O ECR repositório da Amazon deve estar no mesmo Região da AWS que você usa para iniciar aplicativos EMR sem servidor.

Configurando o acesso VPC

Você pode configurar aplicativos EMR sem servidor para se conectar aos seus armazenamentos de dados dentro do seuVPC, como clusters do Amazon Redshift, bancos de dados Amazon ou buckets do RDS Amazon S3 com endpoints. VPC Seu aplicativo EMR sem servidor tem conectividade de saída com os armazenamentos de dados dentro do seu. VPC Por padrão, o EMR Serverless bloqueia o acesso de entrada aos seus aplicativos para melhorar a segurança.

> **Note**
>
> Você deve configurar o VPC acesso se quiser usar um banco de dados externo do Hive Metastore para seu aplicativo. Para obter informações sobre como configurar um metastore externo do Hive, consulte Configuração do Metastore.

Criar aplicativo

Na página Criar aplicativo, você pode escolher configurações personalizadas e especificar as sub-redes e VPC os grupos de segurança que os aplicativos EMR sem servidor podem usar.

VPCs

Escolha o nome da nuvem privada virtual (VPC) que contém seus armazenamentos de dados. A página Criar aplicativo lista todas as VPCs opções escolhidas. Região da AWS.

Sub-redes

Escolha as sub-redes dentro do VPC que contém seu armazenamento de dados. A página Criar aplicativo lista todas as sub-redes dos armazenamentos de dados em seu. VPC

As sub-redes selecionadas devem ser sub-redes privadas. Isso significa que as tabelas de rotas associadas às sub-redes não devem ter gateways de Internet.

Para conectividade de saída com a Internet, as sub-redes devem ter rotas de saída usando um Gateway. NAT Para configurar um NAT gateway, consulte Trabalhar com NAT gateways.

Para conectividade com o Amazon S3, as sub-redes devem ter um NAT gateway ou um endpoint configurado. VPC Para configurar um VPC endpoint S3, consulte Criar um endpoint de gateway.

Para conectividade com outros Serviços da AWS fora doVPC, como o Amazon DynamoDB, você deve configurar endpoints VPC ou um gateway. NAT Para configurar VPC endpoints para Serviços da AWS, consulte Trabalhar com VPC endpoints.

Os trabalhadores podem se conectar aos armazenamentos de dados dentro de você VPC por meio do tráfego de saída. Por padrão, o EMR Serverless bloqueia o acesso de entrada aos trabalhadores para melhorar a segurança.

Quando você usa AWS Config, o EMR Serverless cria um registro de item de interface de rede elástica para cada trabalhador. Para evitar custos relacionados a esse recurso, considere desligar `AWS::EC2::NetworkInterface` AWS Config.

> **Note**
>
> Recomendamos que você selecione várias sub-redes em várias zonas de disponibilidade. Isso ocorre porque as sub-redes que você escolhe determinam as zonas de disponibilidade que estão disponíveis para a inicialização de um EMR aplicativo sem servidor. Cada trabalhador consumirá um endereço IP na sub-rede em que é iniciado. Certifique-se de que as sub-redes especificadas tenham endereços IP suficientes para o número de trabalhadores que você planeja iniciar. Para obter mais informações sobre planejamento de sub-rede, consultethe section called “Práticas recomendadas para planejamento de sub-rede”.

Grupos de segurança

Escolha um ou mais grupos de segurança que possam se comunicar com seus armazenamentos de dados. A página Criar aplicativo lista todos os grupos de segurança em seuVPC. EMRO Serverless associa esses grupos de segurança a interfaces de rede elásticas conectadas às suas sub-redes. VPC

> **Note**
>
> Recomendamos que você crie um grupo de segurança separado para aplicativos EMR sem servidor. Isso torna o isolamento e o gerenciamento das regras de rede mais eficientes. Por exemplo, para se comunicar com os clusters do Amazon Redshift, você pode definir as regras de tráfego entre os grupos de segurança Redshift e EMR Serverless, conforme demonstrado no exemplo abaixo.

Example Exemplo — Comunicação com clusters do Amazon Redshift

1. Adicione uma regra para tráfego de entrada para o grupo de segurança do Amazon Redshift a partir de um dos grupos de segurança EMR Serverless.

Tipo	Protocolo	Intervalo de portas	Origem
Tudo TCP	TCP	5439	`emr-serverless-security-group`

2. Adicione uma regra para o tráfego de saída de um dos grupos de segurança EMR sem servidor. É possível fazer isso de duas formas. Primeiro, você pode abrir o tráfego de saída para todas as portas.

Tipo	Protocolo	Intervalo de portas	Destination (Destino)
Todo o tráfego	TCP	ALL	0.0.0.0/0

Como alternativa, você pode restringir o tráfego de saída para os clusters do Amazon Redshift. Isso é útil somente quando o aplicativo precisa se comunicar com os clusters do Amazon Redshift e nada mais.

Tipo	Protocolo	Intervalo de portas	Origem
Tudo TCP	TCP	5439	`redshift-security-group`

Configurar aplicativo

Você pode alterar a configuração de rede de um aplicativo EMR Serverless existente na página Configurar aplicativo.

Exibir detalhes da execução do trabalho

Na página de detalhes da execução do Job, você pode visualizar a sub-rede usada pelo seu job para uma execução específica. Observe que um trabalho é executado somente em uma sub-rede selecionada das sub-redes especificadas.

Práticas recomendadas para planejamento de sub-rede

AWS os recursos são criados em uma sub-rede que é um subconjunto de endereços IP disponíveis em uma Amazon. VPC Por exemplo, um VPC com uma máscara de rede /16 tem até 65.536 endereços IP disponíveis que podem ser divididos em várias redes menores usando máscaras de sub-rede. Como exemplo, você pode dividir esse intervalo em duas sub-redes, cada uma usando a máscara /17 e 32.768 endereços IP disponíveis. Uma sub-rede reside em uma zona de disponibilidade e não pode se estender entre zonas.

As sub-redes devem ser projetadas tendo em mente os limites de escalabilidade de aplicativos EMR sem servidor. Por exemplo, se você tiver um aplicativo solicitando 4 vCpu trabalhadores e puder escalar até 4.000vCpu, seu aplicativo precisará de no máximo 1.000 trabalhadores para um total de 1.000 interfaces de rede. Recomendamos que você crie sub-redes em várias zonas de disponibilidade. Isso permite que o EMR Serverless repita seu trabalho ou provisione a capacidade pré-inicializada em uma zona de disponibilidade diferente em um evento improvável quando uma zona de disponibilidade falhar. Portanto, cada sub-rede em pelo menos duas zonas de disponibilidade deve ter mais de 1.000 endereços IP disponíveis.

Você precisa de sub-redes com tamanho de máscara menor ou igual a 22 para provisionar 1.000 interfaces de rede. Qualquer máscara maior que 22 não atenderá ao requisito. Por exemplo, uma máscara de sub-rede de /23 fornece 512 endereços IP, enquanto uma máscara de /22 fornece 1024 e uma máscara de /21 fornece 2048 endereços IP. Abaixo está um exemplo de 4 sub-redes com máscara /22 em uma máscara VPC de rede /16 que podem ser alocadas em diferentes zonas de disponibilidade. Há uma diferença de cinco entre endereços IP disponíveis e utilizáveis porque os primeiros quatro endereços IP e o último endereço IP em cada sub-rede são reservados por AWS.

ID da sub-rede	Endereço de sub-rede	Máscara de sub-rede	Intervalo de endereços IP	Endereços IP disponíveis	Endereços IP utilizáveis
1	10.0.0.0	255.252.0/22	10.0.0.0 - 10.0.3.255	1,024	1.019

ID da sub-rede	Endereço de sub-rede	Máscara de sub-rede	Intervalo de endereços IP	Endereços IP disponíveis	Endereços IP utilizáveis
2	10.0.4.0	255.252.0/22	10.0.4.0 - 10.0.7.255	1,024	1.019
3	10.0.8.0	255.252.0/22	10.0.4.0 - 10.0.7.255	1,024	1.019
4	10.0.12.0	255.252.0/22	10.0.12.0 - 10.0.15.255	1,024	1.019

Você deve avaliar se sua carga de trabalho é mais adequada para funcionários maiores. Usar funcionários maiores requer menos interfaces de rede. Por exemplo, usar 16 vCpu trabalhadores com um limite de escalabilidade de aplicativos de 4.000 vCpu exigirá no máximo 250 trabalhadores para um total de 250 endereços IP disponíveis para provisionar interfaces de rede. Você precisa de sub-redes em várias zonas de disponibilidade com tamanho de máscara menor ou igual a 24 para provisionar 250 interfaces de rede. Qualquer tamanho de máscara maior que 24 oferece menos de 250 endereços IP.

Se você compartilha sub-redes em vários aplicativos, cada sub-rede deve ser projetada tendo em mente os limites de escalabilidade coletiva de todos os seus aplicativos. Por exemplo, se você tiver 3 aplicativos solicitando 4 vCpu trabalhadores e cada um puder escalar até 4000 vCpu com uma cota baseada em serviços de 12.000 no vCpu nível da conta, cada sub-rede exigirá 3.000 endereços IP disponíveis. Se o VPC que você deseja usar não tiver um número suficiente de endereços IP, tente aumentar o número de endereços IP disponíveis. Você pode fazer isso associando blocos adicionais de roteamento entre domínios sem classe (CIDR) ao seu. VPC Para obter mais informações, consulte Associar IPv4 CIDR blocos adicionais ao seu VPC no Guia VPC do usuário da Amazon.

Você pode usar uma das muitas ferramentas disponíveis on-line para gerar rapidamente definições de sub-rede e analisar o intervalo disponível de endereços IP.

Opções de EMR arquitetura Amazon Serverless

A arquitetura do conjunto de instruções do seu aplicativo Amazon EMR Serverless determina o tipo de processador que o aplicativo usa para executar o trabalho. EMRA Amazon oferece duas opções de arquitetura para seu aplicativo: x86_64 e arm64. EMRO Serverless atualiza automaticamente a

última geração de instâncias à medida que elas se tornam disponíveis, para que seus aplicativos possam usar as instâncias mais recentes sem exigir esforço adicional de você.

Tópicos

Usando a arquitetura x86_64

A arquitetura x86_64 também é conhecida como x86 de 64 bits ou x64. x86_64 é a opção padrão para EMR aplicativos sem servidor. Essa arquitetura usa processadores baseados em x86 e é compatível com a maioria das ferramentas e bibliotecas de terceiros.

A maioria dos aplicativos é compatível com a plataforma de hardware x86 e pode ser executada com êxito na arquitetura x86_64 padrão. No entanto, se seu aplicativo for compatível com 64 bitsARM, você poderá mudar para arm64 para usar os processadores Graviton para melhorar o desempenho, a potência computacional e a memória. Custa menos executar instâncias na arquitetura arm64 do que quando você executa instâncias do mesmo tamanho na arquitetura x86.

Usando a arquitetura arm64 (Graviton)

AWS Os processadores Graviton são projetados de forma personalizada por AWS com núcleos ARM Neoverse de 64 bits e aproveite a arquitetura arm64 (também conhecida como Arch64 ou 64 bits). ARM A ferramenta AWS A linha de processadores Graviton disponível no EMR Serverless inclui os processadores Graviton3 e Graviton2. Esses processadores oferecem uma relação preço-desempenho superior para cargas de trabalho Spark e Hive em comparação com cargas de trabalho equivalentes que são executadas na arquitetura x86_64. EMRO Serverless usa automaticamente a última geração de processadores quando disponível, sem nenhum esforço de sua parte para fazer o upgrade para a última geração de processadores.

Lançamento de novos aplicativos com suporte ao Graviton

Use um dos métodos a seguir para iniciar um aplicativo que usa a arquitetura arm64.

AWS CLI

Para iniciar um aplicativo usando processadores Graviton do AWS CLI, especifique `ARM64` como `architecture` parâmetro no `create-application`API. Forneça os valores apropriados para seu aplicativo nos outros parâmetros.

```
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

Para iniciar um aplicativo usando os processadores Graviton do EMR Studio, escolha arm64 como a opção Arquitetura ao criar ou atualizar um aplicativo.

Configurando aplicativos existentes para usar o Graviton

Você pode configurar seus aplicativos Amazon EMR Serverless existentes para usar a arquitetura Graviton (arm64) com o, SDK AWS CLI, ou EMR Studio.

Para converter um aplicativo existente de x86 para arm64

1. Confirme se você está usando a versão principal mais recente do AWS CLI/SDKque suporta o `architecture` parâmetro.
2. Confirme se não há trabalhos em execução e, em seguida, interrompa o aplicativo.

   ```
   aws emr-serverless stop-application \
    --application-id application-id \
    --region us-west-2
   ```
3. Para atualizar o aplicativo para usar o Graviton, especifique `ARM64` o `architecture` parâmetro no `update-application`API.

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

4. Para verificar se a CPU arquitetura do aplicativo está agoraARM64, use `get-application` API o.

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

5. Quando estiver pronto, reinicie o aplicativo.

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

Considerações ao usar o Graviton

Antes de iniciar um aplicativo EMR Serverless usando arm64 para suporte ao Graviton, confirme o seguinte.

Compatibilidade de bibliotecas

Ao selecionar Graviton (arm64) como opção de arquitetura, certifique-se de que pacotes e bibliotecas de terceiros sejam compatíveis com a arquitetura de 64 bitsARM. Para obter informações sobre como empacotar bibliotecas Python em um ambiente virtual Python compatível com a arquitetura selecionada, consulte. Usando bibliotecas Python com Serverless EMR

Para saber mais sobre como configurar uma carga de trabalho do Spark ou do Hive para usar 64 bitsARM, consulte o AWSRepositório Graviton Getting Started ativado. GitHub Esse repositório contém recursos essenciais que podem ajudá-lo a começar a usar o Graviton ARM baseado.

Coloque dados no S3 Express One Zone com EMR o Serverless

Com as EMR versões 7.2.0 e superiores da Amazon, você pode usar o EMR Serverless com a classe de armazenamento Amazon S3 Express One Zone para melhorar o desempenho ao executar trabalhos e cargas de trabalho. O S3 Express One Zone é uma classe de armazenamento Amazon S3 de zona única e alto desempenho que fornece acesso consistente a dados de um dígito em milissegundos para a maioria dos aplicativos sensíveis à latência. Na hora da execução, o S3 Express One Zone oferece o armazenamento de objetos na nuvem com a menor latência e a maior performance do Amazon S3.

Pré-requisitos

- Permissões do S3 Express One Zone — Quando o S3 Express One Zone executa inicialmente uma ação como `GETLIST`, ou `PUT` em um objeto do S3, a classe de armazenamento chama `CreateSession` em seu nome. Sua IAM política deve permitir a `s3express:CreateSession` permissão para que o S3A o conector pode invocar o. `CreateSession` API Para obter um exemplo de política com essa permissão, consulte Conceitos básicos da classe S3 Express One Zone.
- S3A conector — Para configurar o Spark para acessar dados de um bucket Amazon S3 que usa a classe de armazenamento S3 Express One Zone, você deve usar o conector Apache Hadoop S3A. Para usar o conector, certifique-se de que todos os S3 URIs usem o `s3a` esquema. Caso contrário, você pode alterar a implementação do sistema de arquivos usado para os esquemas do `s3` e do `s3n`.

Para alterar o esquema do `s3`, especifique as seguintes configurações de cluster:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

Para alterar o esquema do `s3n`, especifique as seguintes configurações de cluster:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

Conceitos básicos da classe S3 Express One Zone

Siga estas etapas para começar a usar o S3 Express One Zone.

1. Crie um VPC endpoint. Adicione o endpoint `com.amazonaws.us-west-2.s3express` ao VPC endpoint.
2. Siga Getting Started with Amazon EMR Serverless para criar um aplicativo com a etiqueta de EMR lançamento da Amazon 7.2.0 ou superior.
3. Configure seu aplicativo para usar o VPC endpoint recém-criado, um grupo de sub-rede privado e um grupo de segurança.
4. Adicione a `CreateSession` permissão à sua função de execução do trabalho.

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

5. Administre seu trabalho. Observe que você deve usar o `S3A` esquema para acessar os buckets One Zone do S3 Express.

```
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
 }'
```

Execução de trabalhos

Depois de provisionar sua inscrição, você pode enviar trabalhos para a inscrição. Esta seção aborda como usar o AWS CLI para executar esses trabalhos. Esta seção também identifica os valores padrão para cada tipo de aplicativo disponível no EMR Serverless.

Tópicos

Estados de execução de trabalho

Quando você envia uma execução de trabalho para uma fila de trabalhos do Amazon EMR Serverless, a execução do trabalho entra no estado. `SUBMITTED` Um estado de trabalho `SUBMITTED` passa de `RUNNING` até atingir`FAILED`,`SUCCESS`, ou`CANCELLING`.

As execuções de trabalhos podem ter os seguintes estados:

Estado	Descrição
Enviado	O estado inicial do trabalho quando você envia uma execução de trabalho para o EMR Serverless. O trabalho espera ser agendado para a inscrição. EMRO Serverless começa a priorizar e programar a execução do trabalho.

Estado	Descrição
Pendente	O programador está avaliando a execução do trabalho para priorizar e programar a execução do aplicativo.
Programado	EMRO Serverless agendou a execução do trabalho para o aplicativo e está alocando recursos para executar o trabalho.
Running	EMRO Serverless alocou os recursos de que o trabalho precisa inicialmente, e o trabalho está sendo executado no aplicativo. Em aplicaçõe s do Spark, isso significa que o processo do driver do Spark está no estado `running`.
Com falha	EMRO Serverless falhou ao enviar a execução da tarefa para o aplicativo ou ela foi concluída sem êxito. Consulte `StateDetails` para obter informações adicionais sobre essa falha no trabalho.
Bem-sucedida	A execução do trabalho foi concluída com êxito.
Cancelando	Ele `CancelJobRun` API solicitou o cancelamento da execução do trabalho ou o tempo limite da execução do trabalho expirou. EMRO Serverless está tentando cancelar o trabalho no aplicativo e liberar os recursos.
Cancelado	A execução do trabalho foi cancelada com êxito e os recursos usados foram liberados.

Executando trabalhos a partir do console do EMR Studio

Você pode enviar execuções de trabalhos para aplicativos EMR sem servidor e visualizar os trabalhos no console do EMR Studio. Para criar ou navegar até seu aplicativo EMR sem servidor no console do EMR Studio, siga as instruções em Introdução ao console.

Enviar um trabalho

Na página Enviar trabalho, você pode enviar um trabalho para um aplicativo EMR sem servidor da seguinte forma.

Spark

1. No campo Nome, insira um nome para a execução do trabalho.
2. No campo Função de tempo de execução, insira o nome da IAM função que seu aplicativo EMR sem servidor pode assumir para a execução do trabalho. Para saber mais sobre as funções de tempo de execução, consulteFunções de tempo de execução de trabalho para Amazon EMR Serverless.
3. No campo Localização do script, insira a localização do Amazon S3 para o script ou JAR que você deseja executar. Para trabalhos do Spark, o script pode ser um arquivo Python `.py` () ou JAR um arquivo `.jar` ().
4. Se a localização do seu script for um JAR arquivo, insira o nome da classe que é o ponto de entrada para o trabalho no campo Classe principal.
5. (Opcional) Insira valores para os campos restantes.

   - Argumentos do script — insira os argumentos que você deseja passar para o script principal JAR ou Python. Seu código lê esses parâmetros. Separe cada argumento na matriz por uma vírgula.
   - Propriedades do Spark — Expanda a seção de propriedades do Spark e insira quaisquer parâmetros de configuração do Spark nesse campo.

     > **Note**
     >
     > Se você especificar os tamanhos do driver e do executor do Spark, deverá considerar a sobrecarga de memória. Especifique os valores de sobrecarga de memória nas propriedades `spark.driver.memoryOverhead` e. `spark.executor.memoryOverhead` A sobrecarga de memória tem um valor

padrão de 10% da memória do contêiner, com um mínimo de 384 MB. Juntas, a memória do executor e a sobrecarga de memória não podem exceder a memória do trabalhador. Por exemplo, o máximo `spark.executor.memory` em um trabalhador de 30 GB deve ser 27 GB.

- Configuração de trabalho — especifique qualquer configuração de trabalho nesse campo. Você pode usar essas configurações de trabalho para substituir as configurações padrão dos aplicativos.
- Configurações adicionais — Ative ou desative o AWS Glue o Data Catalog como um metastore e modifique as configurações de log do aplicativo. Para saber mais sobre as configurações do metastore, consulte. Configuração do Metastore Para saber mais sobre as opções de registro de aplicativos, consulteArmazenando registros.
- Tags — Atribua tags personalizadas ao aplicativo.

6. Escolha Enviar trabalho.

Hive

1. No campo Nome, insira um nome para a execução do trabalho.
2. No campo Função de tempo de execução, insira o nome da IAM função que seu aplicativo EMR sem servidor pode assumir para a execução do trabalho.
3. No campo Localização do script, insira a localização do Amazon S3 para o script ou JAR que você deseja executar. Para trabalhos do Hive, o script deve ser um arquivo Hive (`.sql`).
4. (Opcional) Insira valores para os campos restantes.

   - Localização do script de inicialização — Insira a localização do script que inicializa as tabelas antes da execução do script do Hive.
   - Propriedades do Hive — Expanda a seção Propriedades do Hive e insira quaisquer parâmetros de configuração do Hive nesse campo.
   - Configuração de trabalho — especifique qualquer configuração de trabalho. Você pode usar essas configurações de trabalho para substituir as configurações padrão dos aplicativos. Para trabalhos do Hive, `hive.exec.scratchdir` e `hive.metastore.warehouse.dir` são propriedades obrigatórias na `hive-site` configuração.

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

- Configurações adicionais — Ative ou desative o AWS Glue o Data Catalog como um metastore e modifique as configurações de log do aplicativo. Para saber mais sobre as configurações do metastore, consulte. Configuração do Metastore Para saber mais sobre as opções de registro de aplicativos, consulteArmazenando registros.
- Tags — Atribua quaisquer tags personalizadas ao aplicativo.

5. Escolha Enviar trabalho.

Visualizar execuções de trabalhos

Na guia Execuções de trabalhos na página Detalhes de um aplicativo, você pode visualizar execuções de trabalhos e realizar as seguintes ações para execuções de trabalhos.

Cancelar tarefa — Para cancelar a execução de uma tarefa que esteja no `RUNNING` estado, escolha essa opção. Para saber mais sobre as transições de execução de tarefas, consulteEstados de execução de trabalho.

Clonar tarefa — Para clonar uma execução de tarefa anterior e reenviá-la, escolha essa opção.

Executando trabalhos a partir do AWS CLI

Você pode criar, descrever e excluir trabalhos individuais no AWS CLI. Você também pode listar todos os seus trabalhos para visualizá-los rapidamente.

Para enviar um novo trabalho, use`start-job-run`. Forneça o ID do aplicativo que você deseja executar, junto com as propriedades específicas do trabalho. Para exemplos do Spark,

consulteTrabalhos do Spark. Para exemplos do Hive, consulteTrabalhos na Hive. Este comando retorna seu`application-id`,ARN, e novo`job-id`.

Cada execução de trabalho tem uma duração de tempo limite definida. Se a execução do trabalho exceder essa duração, o EMR Serverless o cancelará automaticamente. O tempo limite padrão é de 12 horas. Ao iniciar a execução do trabalho, você pode definir essa configuração de tempo limite para um valor que atenda aos requisitos do trabalho. Configure o valor com a `executionTimeoutMinutes` propriedade.

```
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://DOC-EXAMPLE-BUCKET/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/warehouse"
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

Para descrever um trabalho, use`get-job-run`. Esse comando retorna as configurações específicas do trabalho e a capacidade definida para seu novo trabalho.

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

Para listar seus trabalhos, use`list-job-runs`. Esse comando retorna um conjunto abreviado de propriedades que inclui tipo de tarefa, estado e outros atributos de alto nível. Se você não quiser ver todos os seus trabalhos, você pode especificar o número máximo de trabalhos que deseja ver, até 50. O exemplo a seguir especifica que você deseja ver seus dois últimos trabalhos executados.

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

Para cancelar um trabalho, usecancel-job-run. Forneça o `application-id` `job-id` e o trabalho que você deseja cancelar.

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

Para obter mais informações sobre como executar trabalhos a partir do AWS CLI, consulte a Referência EMRsem servidor API.

Usando discos otimizados para reprodução aleatória

Com as EMR versões 7.1.0 e superiores da Amazon, você pode usar discos otimizados para reprodução aleatória ao executar trabalhos do Apache Spark ou do Hive para melhorar o desempenho de cargas de trabalho com uso intenso de E/S. Em comparação com os discos padrão, os discos otimizados para reprodução aleatória fornecem mais IOPS (operações de E/S por segundo) para uma movimentação de dados mais rápida e uma latência reduzida durante as operações aleatórias. Os discos otimizados para reprodução aleatória permitem que você conecte tamanhos de disco de até 2 TB por trabalhador, para que você possa configurar a capacidade apropriada para seus requisitos de carga de trabalho.

Benefícios principais

Os discos otimizados para reprodução aleatória oferecem os seguintes benefícios.

- Alto IOPS desempenho — os discos otimizados para reprodução aleatória fornecem discos IOPS mais altos do que os discos padrão, resultando em um embaralhamento de dados mais eficiente e rápido durante tarefas do Spark e do Hive e outras cargas de trabalho intensivas.
- Tamanho de disco maior — os discos otimizados para reprodução aleatória oferecem suporte a tamanhos de disco de 20 GB a 2 TB por trabalhador, para que você possa escolher a capacidade adequada com base em suas cargas de trabalho.

Conceitos básicos

Veja as etapas a seguir para usar discos otimizados para reprodução aleatória em seus fluxos de trabalho.

Spark

1. Crie um aplicativo EMR Serverless versão 7.1.0 com o comando a seguir.

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

2. Configure sua tarefa do Spark para incluir os parâmetros `spark.emr-serverless.driver.disk.type` e/ou para ser executada com `spark.emr-serverless.executor.disk.type` discos otimizados para reprodução aleatória. Você pode usar um ou ambos os parâmetros, dependendo do seu caso de uso.

```
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"
        }
    }'
```

   Para obter mais informações, consulte Propriedades de trabalho do Spark.

Hive

1. Crie um aplicativo EMR Serverless versão 7.1.0 com o comando a seguir.

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

2. Configure sua tarefa do Hive para incluir os parâmetros `hive.driver.disk.type` e/ou para ser executada com `hive.tez.disk.type` discos otimizados para reprodução aleatória. Você pode usar um ou ambos os parâmetros, dependendo do seu caso de uso.

```
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"
            }
        }]
    }'
```

Para obter mais informações, Hive job properties.

Configurando um aplicativo com capacidade pré-inicializada

Veja os exemplos a seguir para criar aplicativos com base na EMR versão 7.1.0 da Amazon. Esses aplicativos têm as seguintes propriedades:

- 5 drivers Spark pré-inicializados, cada um com 2 vCPU, 4 GB de memória e 50 GB de disco otimizado para reprodução aleatória.
- 50 executores pré-inicializados, cada um com 4 vCPU, 8 GB de memória e 500 GB de disco otimizado para reprodução aleatória.

Quando esse aplicativo executa trabalhos do Spark, ele primeiro consome os trabalhadores pré-inicializados e depois escala os trabalhadores sob demanda até a capacidade máxima de 400 v CPU e 1024 GB de memória. Opcionalmente, você pode omitir a capacidade de um ou. `DRIVER` `EXECUTOR`

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"
  }'
```

Trabalhos de streaming

Um trabalho de streaming no EMR Serverless é um modo de trabalho que permite analisar e processar dados de streaming quase em tempo real. Esses trabalhos de longa duração pesquisam dados de streaming e processam continuamente os resultados à medida que os dados chegam. Os trabalhos de streaming são mais adequados para tarefas que exigem processamento de dados em tempo real, como análises quase em tempo real, detecção de fraudes e mecanismos de recomendações. EMRAs tarefas de streaming sem servidor fornecem otimizações, como resiliência de trabalho integrada, monitoramento em tempo real, gerenciamento aprimorado de registros e integração com conectores de streaming.

Veja a seguir alguns casos de uso com trabalhos de streaming:

- Análise quase em tempo real — os trabalhos de streaming no Amazon EMR Serverless permitem que você processe dados de streaming quase em tempo real, para que você possa realizar análises em tempo real em fluxos de dados contínuos, como dados de log, dados de sensores ou dados de sequência de cliques para obter insights e tomar decisões oportunas com base nas informações mais recentes.
- Detecção de fraudes — você pode usar trabalhos de streaming para executar a detecção de fraudes quase em tempo real em transações financeiras, operações de cartão de crédito ou atividades on-line ao analisar fluxos de dados e identificar padrões ou anomalias suspeitas à medida que ocorrem.
- Mecanismos de recomendação — trabalhos de streaming podem processar dados de atividade do usuário e atualizar modelos de recomendações. Isso abre possibilidades para recomendações personalizadas e em tempo real com base em comportamentos e preferências.
- Análise de mídia social — trabalhos de streaming podem processar dados de mídia social, como tweets, comentários e postagens, para que as organizações possam monitorar tendências, analisar sentimentos e gerenciar a reputação da marca quase em tempo real.
- Análise da Internet das Coisas (IoT) — os trabalhos de streaming podem lidar e analisar fluxos de dados de alta velocidade de dispositivos, sensores e máquinas conectadas de IoT, para que você possa executar a detecção de anomalias, a manutenção preditiva e outros casos de uso de análises de IoT.
- Análise de fluxo de cliques — trabalhos de streaming podem processar e analisar dados de fluxo de cliques de sites ou aplicativos móveis. As empresas que usam esses dados podem executar análises para aprender mais sobre o comportamento do usuário, personalizar as experiências do usuário e otimizar as campanhas de marketing.

- Monitoramento e análise de registros — os trabalhos de streaming também podem processar dados de log de servidores, aplicativos e dispositivos de rede. Isso fornece detecção de anomalias, solução de problemas e integridade e desempenho do sistema.

Principais benefícios

Os trabalhos de streaming no EMR Serverless fornecem automaticamente resiliência ao trabalho, que é uma combinação dos seguintes fatores:

- Tentativa automática — O EMR Serverless repete automaticamente todas as tarefas que falharam sem qualquer intervenção manual de sua parte.
- Resiliência da Zona de Disponibilidade (AZ) — O EMR Serverless muda automaticamente as tarefas de streaming para uma AZ saudável se a AZ original apresentar problemas.
- Gerenciamento de registros:
  - Rotação de registros — para um gerenciamento mais eficiente do armazenamento em disco, o EMR Serverless alterna periodicamente os registros para trabalhos de streaming longos. Isso evita o acúmulo de registros que podem consumir todo o espaço em disco.
  - Compactação de registros — ajuda você a gerenciar e otimizar com eficiência os arquivos de log em persistência gerenciada. A compactação também melhora a experiência de depuração quando você usa o servidor gerenciado de histórico do Spark.

Fontes de dados e coletores de dados compatíveis

EMRO Serverless funciona com várias fontes de dados de entrada e coletores de dados de saída:

- Fontes de dados de entrada suportadas — Amazon Kinesis Data Streams, Amazon Managed Streaming for Apache Kafka e clusters autogerenciados do Apache Kafka. Por padrão, as EMR versões 7.1.0 e superiores da Amazon incluem o conector Amazon Kinesis Data Streams, então você não precisa criar ou baixar nenhum pacote adicional.
- Coletores de dados de saída compatíveis — AWS Tabelas do Glue Data Catalog, Amazon S3, Amazon Redshift, SQL My, SQL Postgre Oracle, Oracle, SQL Microsoft, Apache Iceberg, Delta Lake e Apache Hudi.

Considerações e limitações

Ao usar trabalhos de streaming, tenha em mente as seguintes considerações e limitações.

- Os trabalhos de streaming são compatíveis com as EMRversões 7.1.0 e superiores da Amazon.
- EMRO Serverless espera que os trabalhos de streaming sejam executados por muito tempo, então você não pode definir o tempo limite de execução para limitar o tempo de execução do trabalho.
- As tarefas de streaming são compatíveis apenas com o mecanismo Spark, que é construído sobre a estrutura estruturada de streaming.
- EMRO Serverless repete indefinidamente os trabalhos de streaming, e você não pode personalizar o número máximo de tentativas. A prevenção de thrash é incluída automaticamente para interromper a repetição do trabalho se a quantidade de tentativas malsucedidas ultrapassar o limite definido em uma janela horária. O limite padrão é de cinco tentativas malsucedidas em uma hora. Você pode configurar esse limite para estar entre 1 e 10 tentativas. Para obter mais informações, consulte Job resiliency.
- Os trabalhos de streaming têm pontos de verificação para salvar o estado e o progresso do tempo de execução, para que o EMR Serverless possa retomar o trabalho de streaming a partir do ponto de verificação mais recente. Para obter mais informações, consulte Recuperação de falhas com o Checkpoint na documentação do Apache Spark.

Introdução aos trabalhos de streaming

Veja as instruções a seguir para saber como começar a usar trabalhos de streaming.

1. Siga Getting Started with Amazon EMR Serverless para criar um aplicativo. Observe que seu aplicativo deve executar a EMRversão 7.1.0 ou superior da Amazon.
2. Quando seu aplicativo estiver pronto, defina o `mode` parâmetro `STREAMING` para enviar um trabalho de streaming, semelhante ao seguinte AWS CLI exemplo.

```
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"
```

```
    }
}'
```

Conectores de streaming compatíveis

Os conectores de streaming facilitam a leitura de dados de uma fonte de streaming e também podem gravar dados em um coletor de streaming.

A seguir estão os conectores de streaming compatíveis:

Conector Amazon Kinesis Data Streams

O conector Amazon Kinesis Data Streams para Apache Spark permite criar aplicativos e pipelines de streaming que consomem dados e gravam dados no Amazon Kinesis Data Streams. O conector suporta um consumo aprimorado de fan-out com uma taxa de transferência de leitura dedicada de até 2 MB/segundo por fragmento. Por padrão, o Amazon EMR Serverless 7.1.0 e versões posteriores incluem o conector, então você não precisa criar ou baixar nenhum pacote adicional. Para obter mais informações sobre o conector, consulte a spark-sql-kinesis-connector página em GitHub.

Veja a seguir um exemplo de como iniciar a execução de um trabalho com a dependência do conector do 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"
    }
}'
```

Para se conectar ao Kinesis Data Streams, você deve EMR configurar o VPC aplicativo Serverless com acesso e VPC usar um endpoint para permitir acesso privado ou NAT usar um Gateway para obter acesso público. Para obter mais informações, consulte Configurando o VPC acesso. Você também deve garantir que sua função de tempo de execução de trabalho tenha as permissões de leitura e gravação necessárias para acessar os fluxos de dados necessários. Para saber mais sobre como configurar uma função de tempo de execução de trabalho, consulte Funções de tempo de execução de tarefas para Amazon EMR Serverless. Para obter uma lista completa de todas as permissões necessárias, consulte a spark-sql-kinesis-connector página em GitHub.

Conector Apache Kafka

O conector Apache Kafka para streaming estruturado do Spark é um conector de código aberto da comunidade Spark e está disponível em um repositório Maven. Esse conector facilita que os aplicativos de streaming estruturados do Spark leiam e gravem dados no Apache Kafka autogerenciado e no Amazon Managed Streaming for Apache Kafka. Para obter mais informações sobre o conector, consulte o Guia de integração do Structured Streaming + Kafka na documentação do Apache Spark.

O exemplo a seguir demonstra como incluir o conector Kafka em sua solicitação de execução de trabalho.

```
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>"
    }
}'
```

A versão do conector Apache Kafka depende da sua versão EMR sem servidor e da versão correspondente do Spark. Para encontrar a versão correta do Kafka, consulte o Guia de integração Structured Streaming + Kafka.

Para usar o Amazon Managed Streaming for Apache IAM Kafka com autenticação, você deve incluir outra dependência para permitir que o conector Kafka se conecte à Amazon. MSK IAM Para obter mais informações, consulte o aws-msk-iam-auth repositório em. GitHub Você também deve se certificar de que a função de tempo de execução do trabalho tenha as IAM permissões necessárias. O exemplo a seguir demonstra como usar o conector com IAM autenticação.

```
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>"
    }
}'
```

Para usar o conector Kafka e a biblioteca de IAM autenticação da Amazon, MSK você deve configurar o aplicativo EMR Serverless com acesso. VPC Suas sub-redes devem ter acesso à Internet e usar um NAT Gateway para acessar as dependências do Maven. Para obter mais informações, consulte Configurando o VPC acesso. As sub-redes devem ter conectividade de rede para acessar o cluster Kafka. Isso é verdade independentemente de seu cluster Kafka ser autogerenciado ou se você usa o Amazon Managed Streaming for Apache Kafka.

Gerenciamento de registros de tarefas de streaming

Rotação de toras

Os trabalhos de streaming oferecem suporte à rotação de registros de aplicativos e registros de eventos do Spark. A rotação de registros impede que trabalhos de streaming longos gerem grandes arquivos de log que podem ocupar todo o espaço disponível em disco. A rotação de registros ajuda a economizar armazenamento em disco e evita falhas no trabalho devido ao pouco espaço em disco. Para obter mais informações, consulte Rotação de registros.

Compactação de toras

Os trabalhos de streaming também oferecem suporte à compactação de registros de eventos do Spark sempre que o registro gerenciado estiver disponível. Para obter mais detalhes sobre o registro gerenciado, consulte Registro com armazenamento gerenciado. Os trabalhos de streaming podem ser executados por um longo tempo, e a quantidade de dados de eventos pode se acumular com o tempo e aumentar significativamente o tamanho dos arquivos de log. O Spark History Server lê e carrega esses eventos na memória para a interface do usuário do aplicativo Spark. Esse processo pode causar altas latências e custos, especialmente se os registros de eventos armazenados no Amazon S3 forem muito grandes.

A compactação de registros reduz o tamanho do registro de eventos, então o Spark History Server não precisa carregar mais de 1 GB de registros de eventos a qualquer momento. Para obter mais informações, consulte Monitoramento e instrumentação na documentação do Apache Spark.

Trabalhos do Spark

Você pode executar trabalhos do Spark em um aplicativo com o `type` parâmetro definido como. `SPARK` Os trabalhos devem ser compatíveis com a versão do Spark compatível com a versão de EMR lançamento da Amazon. Por exemplo, quando você executa trabalhos com a EMR versão 6.6.0 da Amazon, seu trabalho deve ser compatível com o Apache Spark 3.2.0. Para obter informações sobre as versões do aplicativo para cada versão, consulteVersões de EMR lançamento do Amazon Serverless.

Parâmetros do trabalho do Spark

Ao usar o `StartJobRun`APIpara executar uma tarefa do Spark, você pode especificar os seguintes parâmetros.

Parâmetros necessários

- Função de tempo de execução do trabalho do Spark
- Parâmetro do driver de tarefas do Spark

- Parâmetro de substituição da configuração do Spark
- Otimização dinâmica da alocação de recursos do Spark

Função de tempo de execução do trabalho do Spark

Use **`executionRoleArn`**para especificar a ARN IAM função que seu aplicativo usa para executar trabalhos do Spark. Essa função deve conter as seguintes permissões:

- Leia a partir de buckets do S3 ou de outras fontes de dados em que seus dados residem
- Leia dos buckets ou prefixos do S3 em que seu PySpark script ou arquivo reside JAR
- Grave nos buckets do S3 onde você pretende gravar sua saída final
- Grave registros em um bucket ou prefixo do S3 que especifique `S3MonitoringConfiguration`
- Acesso às KMS chaves se você usar KMS chaves para criptografar dados em seu bucket do S3
- Acesso ao AWS Glue Data Catalog se você usa o Spark SQL

Se seu trabalho do Spark ler ou gravar dados em ou de outras fontes de dados, especifique as permissões apropriadas nessa IAM função. Se você não fornecer essas permissões para a IAM função, o trabalho poderá falhar. Para ter mais informações, consulte Funções de tempo de execução de trabalho para Amazon EMR Serverless e Armazenando registros.

Parâmetro do driver de tarefas do Spark

Use **`jobDriver`**para fornecer informações para o trabalho. O parâmetro do driver de trabalho aceita somente um valor para o tipo de trabalho que você deseja executar. Para uma tarefa do Spark, o valor do parâmetro é `sparkSubmit`. Você pode usar esse tipo de tarefa para executar Scala, Java PySpark, SparkR e qualquer outra tarefa compatível por meio do envio do Spark. As tarefas do Spark têm os seguintes parâmetros:

- **`sparkSubmitParameters`**— Esses são os parâmetros adicionais do Spark que você deseja enviar para o trabalho. Use esse parâmetro para substituir as propriedades padrão do Spark, como memória do driver ou número de executores, como aquelas definidas nos `--conf` argumentos ou. `--class`
- **`entryPointArguments`**— Essa é uma matriz de argumentos que você deseja passar para seu arquivo principal JAR ou Python. Você deve realizar a leitura desses parâmetros usando seu código de Entrypoint. Separe cada argumento na matriz por uma vírgula.

- **`entryPoint`**— Essa é a referência no Amazon S3 para o arquivo principal ou JAR Python que você deseja executar. Se você estiver executando um Scala ou JavaJAR, especifique a classe de entrada principal no `SparkSubmitParameters` usando o `--class` argumento.

Para obter informações adicionais, consulte Launching Applications with spark-submit.

Parâmetro de substituição da configuração do Spark

Use **`configurationOverrides`**para substituir as propriedades de configuração no nível de monitoramento e no nível do aplicativo. Esse parâmetro aceita um JSON objeto com os dois campos a seguir:

- **`monitoringConfiguration`**- Use esse campo para especificar o Amazon S3 URL (`s3MonitoringConfiguration`) em que você deseja que o trabalho EMR sem servidor armazene os registros do seu trabalho do Spark. Certifique-se de ter criado esse bucket com o mesmo Conta da AWS que hospeda seu aplicativo e no mesmo Região da AWS onde seu trabalho está sendo executado.
- **`applicationConfiguration`**— Para substituir as configurações padrão dos aplicativos, você pode fornecer um objeto de configuração nesse campo. Você pode usar uma sintaxe abreviada para fornecer a configuração ou pode referenciar o objeto de configuração em um arquivo. JSON Os objetos de configuração consistem em uma classificação, propriedades e configurações opcionais aninhadas As propriedades consistem nas configurações que você deseja substituir neste arquivo. Você pode especificar várias classificações para vários aplicativos em um único JSON objeto.

> **Note**
>
> As classificações de configuração disponíveis variam de acordo com a versão específica do EMR Serverless. Por exemplo, classificações para Log4j personalizado `spark-executor-log4j2` estão disponíveis somente nas versões 6.8.0 `spark-driver-log4j2` e superiores.

Se você usar a mesma configuração em uma substituição de aplicativo e nos parâmetros de envio do Spark, os parâmetros de envio do Spark terão prioridade. As configurações são classificadas em prioridade da seguinte forma, da mais alta para a mais baixa:

- Configuração que o EMR Serverless fornece quando é criado. `SparkSession`

- Configuração que você fornece como parte `sparkSubmitParameters` do `--conf` argumento.
- A configuração que você fornece como parte do seu aplicativo é substituída quando você inicia um trabalho.
- Configuração que você fornece como parte da sua `runtimeConfiguration` ao criar um aplicativo.
- Configurações otimizadas que a Amazon EMR usa para o lançamento.
- Configurações padrão de código aberto para a aplicação.

Para obter mais informações sobre como declarar configurações no nível do aplicativo e substituir configurações durante a execução do trabalho, consulte. Configuração padrão do aplicativo para EMR Serverless

Otimização dinâmica da alocação de recursos do Spark

Use `dynamicAllocationOptimization` para otimizar o uso de recursos no EMR Serverless. Definir essa propriedade como `true` em sua classificação de configuração do Spark indica que o EMR Serverless otimizará a alocação de recursos do executor para melhor alinhar a taxa na qual o Spark solicita e cancela os executores com a taxa na qual o Serverless cria e libera trabalhadores. EMR Ao fazer isso, o EMR Serverless reutiliza de forma mais otimizada os trabalhadores em todos os estágios, resultando em menor custo ao executar trabalhos com vários estágios, mantendo o mesmo desempenho.

Essa propriedade está disponível em todas as versões de EMR lançamento da Amazon.

A seguir está um exemplo de classificação de configuração com`dynamicAllocationOptimization`.

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

Considere o seguinte se você estiver usando a otimização de alocação dinâmica:

- Essa otimização está disponível para os trabalhos do Spark para os quais você ativou a alocação dinâmica de recursos.
- Para obter a melhor eficiência de custos, recomendamos configurar um limite superior de escalabilidade para os trabalhadores usando a configuração de nível de trabalho `spark.dynamicAllocation.maxExecutors` ou a configuração de capacidade máxima em nível de aplicativo com base em sua carga de trabalho.
- Talvez você não veja uma melhora de custo em trabalhos mais simples. Por exemplo, se seu trabalho for executado em um pequeno conjunto de dados ou terminar de ser executado em um estágio, talvez o Spark não precise de um número maior de executores ou de vários eventos de escalabilidade.
- Trabalhos com uma sequência de um estágio grande, estágios menores e, em seguida, um estágio grande novamente podem sofrer regressão no tempo de execução do trabalho. Como o EMR Serverless usa os recursos com mais eficiência, isso pode levar a menos trabalhadores disponíveis para estágios maiores, levando a um tempo de execução mais longo.

Propriedades de trabalho no Spark

A tabela a seguir lista as propriedades opcionais do Spark e seus valores padrão que você pode substituir ao enviar um trabalho do Spark.

Chave	Descrição	Valor padrão
`spark.archives`	Uma lista de arquivos separados por vírgulas que o Spark extrai no diretório de trabalho de cada executor. Os tipos de arquivo compatíveis incluem `.jar .tar.gz`, `.tgz .zip` e. Para especificar o nome do diretório a ser extraído, adicione # após o nome do arquivo que você deseja extrair. Por exemplo, `file.zip#directory` .	`NULL`

Chave	Descrição	Valor padrão
`spark.authenticate`	Opção que ativa a autenticação das conexões internas do Spark.	`TRUE`
`spark.driver.cores`	O número de núcleos que o driver usa.	4
`spark.driver.extraJavaOptions`	Opções extras de Java para o driver Spark.	`NULL`
`spark.driver.memory`	A quantidade de memória que o driver usa.	14G
`spark.dynamicAllocation.enabled`	Opção que ativa a alocação dinâmica de recursos. Essa opção aumenta ou diminui o número de executores registrados no aplicativo, com base na carga de trabalho.	`TRUE`
`spark.dynamicAllocation.executorIdleTimeout`	O tempo que um executor pode permanecer inativo antes que o Spark o remova. Isso só se aplica se você ativar a alocação dinâmica.	Anos 60
`spark.dynamicAllocation.initialExecutors`	O número inicial de executores a serem executados se você ativar a alocação dinâmica.	`3`
`spark.dynamicAllocation.maxExecutors`	O limite superior para o número de executores se você ativar a alocação dinâmica.	Para 6.10.0 e superior, `infinity` Para 6.9.0 e inferior, `100`

Chave	Descrição	Valor padrão
`spark.dynamicAllocation.minExecutors`	O limite inferior para o número de executores se você ativar a alocação dinâmica.	`0`
`spark.emr-serverless.allocation.batch.size`	O número de contêineres a serem solicitados em cada ciclo de alocação do executor. Há uma lacuna de um segundo entre cada ciclo de alocação.	20
`spark.emr-serverless.driver.disk`	O disco do driver Spark.	20G
`spark.emr-serverless.driverEnv.` *`[KEY]`*	Opção que adiciona variáveis de ambiente ao driver do Spark.	NULL
`spark.emr-serverless.executor.disk`	O disco executor do Spark.	20G
`spark.emr-serverless.memoryOverheadFactor`	Define a sobrecarga de memória a ser adicionada à memória do contêiner do driver e do executor.	0.1
`spark.emr-serverless.driver.disk.type`	O tipo de disco conectado ao driver Spark.	Padrão
`spark.emr-serverless.executor.disk.type`	O tipo de disco conectado aos executores do Spark.	Padrão
`spark.executor.cores`	O número de núcleos que cada executor usa.	4

Chave	Descrição	Valor padrão
`spark.executor.extraJavaOptions`	Opções extras de Java para o executor do Spark.	`NULL`
`spark.executor.instances`	O número de contêineres do executor Spark a serem alocados.	3
`spark.executor.memory`	A quantidade de memória que cada executor usa.	14G
`spark.executorEnv.` *`[KEY]`*	Opção que adiciona variáveis de ambiente aos executores do Spark.	`NULL`
`spark.files`	Uma lista de arquivos separados por vírgula para ir para o diretório de trabalho de cada executor. Você pode acessar os caminhos desses arquivos no executor com`SparkFiles.get(` *`fileName`*`)`.	`NULL`
`spark.hadoop.hive.metastore.client.factory.class`	A classe de implementação da metastore Hive.	`NULL`
`spark.jars`	Jars adicionais para adicionar ao classpath de tempo de execução do driver e dos executores.	`NULL`

Chave	Descrição	Valor padrão
`spark.network.crypto.enabled`	Opção que ativa a RPC criptografia AES baseada. Isso inclui o protocolo de autenticação adicionado no Spark 2.2.0.	`FALSE`
`spark.sql.warehouse.dir`	O local padrão para bancos de dados e tabelas gerenciados.	O valor de `$PWD/spark-warehouse`
`spark.submit.pyFiles`	Uma lista separada por vírgula de `.zip`, `.egg`, ou `.py` arquivos para colocar nos aplicativos `PYTHONPATH` Python.	`NULL`

A tabela a seguir lista os parâmetros padrão de envio do Spark.

Chave	Descrição	Valor padrão
`archives`	Uma lista de arquivos separados por vírgulas que o Spark extrai no diretório de trabalho de cada executor.	`NULL`
`class`	A classe principal do aplicativo (para aplicativos Java e Scala).	`NULL`
`conf`	Uma propriedade arbitrária de configuração do Spark.	`NULL`
`driver-cores`	O número de núcleos que o driver usa.	4

Chave	Descrição	Valor padrão
`driver-memory`	A quantidade de memória que o driver usa.	14G
`executor-cores`	O número de núcleos que cada executor usa.	4
`executor-memory`	A quantidade de memória que o executor usa.	14G
`files`	Uma lista de arquivos separados por vírgula para colocar no diretório de trabalho de cada executor. Você pode acessar os caminhos desses arquivos no executor com`SparkFiles.get( fileName)`.	`NULL`
`jars`	Uma lista de jars separados por vírgulas para incluir nos caminhos de classe do driver e do executor.	`NULL`
`num-executors`	O número de executores a serem lançados.	3
`py-files`	Uma lista separada por vírgulas de`.zip`,`.egg`, ou `.py` arquivos para colocar nos aplicativos `PYTHONPATH` Python.	`NULL`
`verbose`	Opção que ativa a saída de depuração adicional.	`NULL`

Exemplos do Spark

O exemplo a seguir mostra como usar o `StartJobRun` API para executar um script Python. Para obter um end-to-end tutorial que usa esse exemplo, consulteComeçando a usar o Amazon EMR Serverless. Você pode encontrar outros exemplos de como executar PySpark trabalhos e adicionar dependências do Python no repositório EMRServerless 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://DOC-EXAMPLE-BUCKET-OUTPUT/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"
        }
    }'
```

O exemplo a seguir mostra como usar o `StartJobRun` API para executar um SparkJAR.

```
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"
        }
    }'
```

Trabalhos na Hive

Você pode executar trabalhos do Hive em um aplicativo com o `type` parâmetro definido como. `HIVE` Os trabalhos devem ser compatíveis com a versão do Hive compatível com a versão de EMR

lançamento da Amazon. Por exemplo, quando você executa trabalhos em um aplicativo com a EMR versão 6.6.0 da Amazon, seu trabalho deve ser compatível com o Apache Hive 3.1.2. Para obter informações sobre as versões do aplicativo para cada versão, consulteVersões de EMR lançamento do Amazon Serverless.

Parâmetros de trabalho do Hive

Ao usar o `StartJobRun`APIpara executar uma tarefa do Hive, você deve especificar os seguintes parâmetros.

Parâmetros necessários

- Função de tempo de execução do trabalho do Hive
- Parâmetro do driver de trabalho do Hive
- Parâmetro de substituição da configuração do Hive

Função de tempo de execução do trabalho do Hive

Use **`executionRoleArn`**para especificar a ARN IAM função que seu aplicativo usa para executar trabalhos do Hive. Essa função deve conter as seguintes permissões:

- Leia a partir de buckets do S3 ou de outras fontes de dados em que seus dados residem
- Leia os buckets ou prefixos do S3 em que residem o arquivo de consulta do Hive e o arquivo de consulta inicial
- Leia e grave nos buckets do S3 onde residem o diretório do Hive Scratch e o diretório do armazém do Hive Metastore
- Grave nos buckets do S3 onde você pretende gravar sua saída final
- Grave registros em um bucket ou prefixo do S3 que especifique `S3MonitoringConfiguration`
- Acesso às KMS chaves se você usar KMS chaves para criptografar dados em seu bucket do S3
- Acesso ao AWS Glue Data Catalog

Se sua tarefa do Hive ler ou gravar dados em ou de outras fontes de dados, especifique as permissões apropriadas nessa IAM função. Se você não fornecer essas permissões para a IAM função, seu trabalho poderá falhar. Para obter mais informações, consulte Funções de tempo de execução de trabalho para Amazon EMR Serverless.

Parâmetro do driver de trabalho do Hive

Use **`jobDriver`**para fornecer informações para o trabalho. O parâmetro do driver de trabalho aceita somente um valor para o tipo de trabalho que você deseja executar. Quando você especifica `hive` como o tipo de tarefa, o EMR Serverless passa uma consulta do Hive para o parâmetro. `jobDriver` Os trabalhos do Hive têm os seguintes parâmetros:

- **`query`**— Essa é a referência no Amazon S3 ao arquivo de consulta do Hive que você deseja executar.
- **`parameters`**— Essas são as propriedades adicionais de configuração do Hive que você deseja substituir. Para substituir propriedades, passe-as para esse parâmetro como`--hiveconf` *`property=value`*. Para substituir variáveis, passe-as para esse parâmetro como`--hivevar` *`key=value`*.
- **`initQueryFile`**— Este é o arquivo de consulta init do Hive. O Hive executa esse arquivo antes da consulta e pode usá-lo para inicializar tabelas.

Parâmetro de substituição da configuração do Hive

Use **`configurationOverrides`**para substituir as propriedades de configuração no nível de monitoramento e no nível do aplicativo. Esses parâmetros aceitam um JSON objeto com os dois campos a seguir:

- **`monitoringConfiguration`**— Use esse campo para especificar o Amazon S3 URL (`s3MonitoringConfiguration`) em que você deseja que o trabalho EMR sem servidor armazene os registros do seu trabalho do Hive. Certifique-se de criar esse bucket com o mesmo Conta da AWS que hospeda seu aplicativo, e no mesmo Região da AWS onde seu trabalho está sendo executado.
- **`applicationConfiguration`**— Você pode fornecer um objeto de configuração nesse campo para substituir as configurações padrão dos aplicativos. Você pode usar uma sintaxe abreviada para fornecer a configuração ou fazer referência ao objeto de configuração em um arquivo. JSON Os objetos de configuração consistem em uma classificação, propriedades e configurações opcionais aninhadas As propriedades consistem nas configurações que você deseja substituir neste arquivo. Você pode especificar várias classificações para vários aplicativos em um único JSON objeto.

> **Note**
>
> As classificações de configuração disponíveis variam de acordo com a versão específica do EMR Serverless. Por exemplo, classificações para Log4j personalizado `spark-executor-log4j2` estão disponíveis somente nas versões 6.8.0 `spark-driver-log4j2` e superiores.

Se você passar a mesma configuração em uma substituição de aplicativo e nos parâmetros do Hive, os parâmetros do Hive terão prioridade. A lista a seguir classifica as configurações da prioridade mais alta para a prioridade mais baixa.

- Configuração que você fornece como parte dos parâmetros do Hive. `--hiveconf` *`property=value`*
- A configuração que você fornece como parte do seu aplicativo é substituída quando você inicia um trabalho.
- Configuração que você fornece como parte da sua `runtimeConfiguration` ao criar um aplicativo.
- Configurações otimizadas que a Amazon EMR atribui para o lançamento.
- Configurações padrão de código aberto para o aplicativo.

Para obter mais informações sobre como declarar configurações no nível do aplicativo e substituir configurações durante a execução do trabalho, consulte. Configuração padrão do aplicativo para EMR Serverless

Propriedades de trabalho em Hive

A tabela a seguir lista as propriedades obrigatórias que você deve configurar ao enviar uma tarefa do Hive.

Configuração	Descrição
`hive.exec.scratchdir`	O local do Amazon S3 onde o EMR Serverless cria arquivos temporários durante a execução do trabalho do Hive.

Configuração	Descrição
`hive.metastore.warehouse.dir`	A localização dos bancos de dados no Amazon S3 para tabelas gerenciadas no Hive.

A tabela a seguir lista as propriedades opcionais do Hive e seus valores padrão que você pode substituir ao enviar um trabalho do Hive.

Configuração	Descrição	Valor padrão
`fs.s3.customAWSCredentialsProvider`	A ferramenta AWS Provedor de credenciais que você deseja usar.	com.amazonaws.auth.d efaultAWSCredentialsProviderChain
`fs.s3a.aws.credentials.provider`	A ferramenta AWS Provedor de credenciais que você deseja usar com um sistema de arquivos S3A.	com.amazonaws.auth.d efaultAWSCredentialsProviderChain
`hive.auto.convert.join`	Opção que ativa a conversão automática de junções comuns em mapjoins, com base no tamanho do arquivo de entrada.	`TRUE`
`hive.auto.convert.join.noconditionaltask`	Opção que ativa a otimizaçã o quando o Hive converte uma junção comum em uma junção de mapa com base no tamanho do arquivo de entrada.	`TRUE`
`hive.auto.convert.join.noconditionaltask.size`	Uma junção é convertida diretamente em uma junção de mapa abaixo desse tamanho.	O valor ideal é calculado com base na memória de tarefas Tez

Configuração	Descrição	Valor padrão
`hive.cbo.enable`	Opção que ativa otimizações baseadas em custos com a estrutura Calcite.	`TRUE`
`hive.cli.tez.session.async`	Opção para iniciar uma sessão Tez em segundo plano enquanto sua consulta do Hive é compilada. Quando definido como`false`, o Tez AM é iniciado após a compilação da consulta do Hive.	`TRUE`
`hive.compute.query.using.stats`	Opção que ativa o Hive para responder a determinadas consultas com estatísticas armazenadas na metastore . Para estatísticas básicas, `hive.stats.autogather` defina como`TRUE`. Para uma coleção mais avançada de consultas, execute`analyze table queries`.	`TRUE`
`hive.default.fileformat`	O formato de arquivo padrão para `CREATE TABLE` declarações. Você pode substituir isso explicitamente se especificar `STORED AS [FORMAT]` em seu `CREATE TABLE` comando.	`TEXTFILE`

Configuração	Descrição	Valor padrão
`hive.driver.cores`	O número de núcleos a serem usados no processo de driver do Hive.	2
`hive.driver.disk`	O tamanho do disco do driver do Hive.	20G
`hive.driver.disk.type`	O tipo de disco do driver do Hive.	Padrão
`hive.tez.disk.type`	O tamanho do disco para os trabalhadores do chá.	Padrão
`hive.driver.memory`	A quantidade de memória a ser usada por processo do driver do Hive. O Hive CLI e o Tez Application Master compartilham essa memória igualmente com 20% do espaço livre.	6G
`hive.emr-serverless.launch.env.[ KEY]`	Opção para definir a variável de *KEY* ambiente em todos os processos específicos do Hive, como seu driver do Hive, Tez AM e tarefa Tez.	
`hive.exec.dynamic.partition`	Opções que ativam partições dinâmicas emDML/DDL.	`TRUE`

Configuração	Descrição	Valor padrão
`hive.exec.dynamic.partition.mode`	Opção que especifica se você deseja usar o modo estrito ou o modo não estrito. No modo estrito, você deve especificar pelo menos uma partição estática caso você acidentalmente sobrescreva todas as partições. No modo não estrito, todas as partições podem ser dinâmicas.	`strict`
`hive.exec.max.dynamic.partitions`	O número máximo de partições dinâmicas que o Hive cria no total.	1000
`hive.exec.max.dynamic.partitions.pernode`	Número máximo de partições dinâmicas que o Hive cria em cada nó mapeador e redutor.	100
`hive.exec.orc.split.strategy`	Espera um dos seguintes valores:`BI`,`ETL`, ou`HYBRID`. Essa não é uma configuração em nível de usuário. `BI`especifica que você deseja gastar menos tempo na geração dividida do que na execução da consulta. `ETL`especifica que você deseja passar mais tempo na geração dividida. `HYBRID`Especifica uma escolha das estratégias acima com base na heurística.	`HYBRID`

Configuração	Descrição	Valor padrão
`hive.exec.reducers.bytes.per.reducer`	O tamanho por redutor. O padrão é 256 MB. Se o tamanho da entrada for 1G, o trabalho usa 4 redutores.	256000000
`hive.exec.reducers.max`	O número máximo de redutores.	256
`hive.exec.stagingdir`	O nome do diretório que armazena os arquivos temporários que o Hive cria dentro dos locais das tabelas e no local do diretório de rascunho especificado na `hive.exec.scratchdir` propriedade.	`.hive-staging`
`hive.fetch.task.conversion`	Espera um dos seguintes valores:`NONE`,`MINIMAL`, ou`MORE`. O Hive pode converter consultas selecionadas em uma única `FETCH` tarefa. Isso minimiza a latência.	`MORE`
`hive.groupby.position.alias`	Opção que faz com que o Hive use um alias de posição da coluna em `GROUP BY` declarações.	`FALSE`
`hive.input.format`	O formato de entrada padrão. Defina como `HiveInputFormat` se você encontrar problemas com`CombineHiveInputFormat` .	`org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`

Configuração	Descrição	Valor padrão
`hive.log.explain.output`	Opção que ativa explicaçõ es sobre a saída estendida para qualquer consulta no seu registro do Hive.	`FALSE`
`hive.log.level`	O nível de registro do Hive.	`INFO`
`hive.mapred.reduce.tasks.speculative.execution`	Opção que ativa o lançament o especulativo para redutores . Compatível somente com o Amazon EMR 6.10.x e versões anteriores.	`TRUE`
`hive.max-task-containers`	O número máximo de contêineres simultâneos. A memória do mapeador configurada é multiplicada por esse valor para determinar a memória disponível que divide o uso da computação e da preempção de tarefas.	1000
`hive.merge.mapfiles`	Opção que faz com que arquivos pequenos sejam mesclados no final de um trabalho somente de mapa.	`TRUE`
`hive.merge.size.per.task`	O tamanho dos arquivos mesclados no final do trabalho.	256000000
`hive.merge.tezfiles`	Opção que ativa a mesclagem de pequenos arquivos no final de um DAG Tez.	`FALSE`

Configuração	Descrição	Valor padrão
`hive.metastore.client.factory.class`	O nome da classe de fábrica que produz objetos que implementam a `IMetaStoreClient` interface.	`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`
`hive.metastore.glue.catalogid`	Se o AWS O Glue Data Catalog atua como um metastore, mas é executado em um sistema diferente Conta da AWS do que os trabalhos, o ID do Conta da AWS onde os trabalhos estão sendo executados.	`NULL`
`hive.metastore.uris`	A economia URI que o cliente do metastore usa para se conectar ao metastore remoto.	`NULL`
`hive.optimize.ppd`	Opção que ativa a redução do predicado.	`TRUE`
`hive.optimize.ppd.storage`	Opção que ativa o envio de predicados para manipuladores de armazenamento.	`TRUE`
`hive.orderby.position.alias`	Opção que faz com que o Hive use um alias de posição da coluna em `ORDER BY` declarações.	`TRUE`
`hive.prewarm.enabled`	Opção que ativa o pré-aquecimento do recipiente para Tez.	`FALSE`
`hive.prewarm.numcontainers`	O número de recipientes a serem pré-aquecidos para Tez.	10

Configuração	Descrição	Valor padrão
`hive.stats.autogather`	Opção que faz com que o Hive colete estatísticas básicas automaticamente durante o `INSERT OVERWRITE` comando.	`TRUE`
`hive.stats.fetch.column.stats`	Opção que desativa a busca de estatísticas de colunas do metastore. Buscar estatísticas de colunas pode ser caro quando o número de colunas é alto.	`FALSE`
`hive.stats.gather.num.threads`	O número de threads que os comandos `partialscan` and `noscan` analyze usam para tabelas particionadas. Isso se aplica somente aos formatos de arquivo que implementam `StatsProvidingRecordReader` (comoORC).	10
`hive.strict.checks.cartesian.product`	Opções que ativam verificações estritas de junção cartesiana. Essas verificações não permitem um produto cartesiano (uma junção cruzada).	`FALSE`
`hive.strict.checks.type.safety`	Opção que ativa verificações de segurança de tipo estritas e desativa a comparação `bigint` com ambas `string` `double` e.	`TRUE`

Configuração	Descrição	Valor padrão
`hive.support.quoted.identifiers`	Espera um valor de `NONE` ou`COLUMN`. `NONE`implica que somente caracteres alfanuméricos e sublinhados sejam válidos nos identificadores. `COLUMN`implica que os nomes das colunas podem conter qualquer caractere.	`COLUMN`
`hive.tez.auto.reducer.parallelism`	Opção que ativa o recurso de paralelismo do redutor automático Tez. O Hive ainda estima os tamanhos dos dados e define estimativas de paralelismo. O Tez coleta amostras dos tamanhos de saída dos vértices de origem e ajusta as estimativas em tempo de execução conforme necessário.	`TRUE`
`hive.tez.container.size`	A quantidade de memória a ser usada por processo de tarefa Tez.	6144
`hive.tez.cpu.vcores`	O número de núcleos a serem usados para cada tarefa do Tez.	2
`hive.tez.disk.size`	O tamanho do disco para cada contêiner de tarefas.	20G
`hive.tez.input.format`	O formato de entrada para geração de divisões no Tez AM.	`org.apache.hadoop.hive.ql.io.HiveInputFormat`

Configuração	Descrição	Valor padrão
`hive.tez.min.partition.factor`	Limite inferior de redutores que Tez especifica quando você ativa o paralelismo do redutor automático.	0.25
`hive.vectorized.execution.enabled`	Opção que ativa o modo vetorizado de execução da consulta.	`TRUE`
`hive.vectorized.execution.reduce.enabled`	Opção que ativa o modo vetorizado do lado reduzido da execução de uma consulta.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	O nome da classe do driver para uma JDBC metastore.	`org.apache.derby.jdbc.EmbeddedDriver`
`javax.jdo.option.ConnectionPassword`	A senha associada a um banco de dados do metastore.	`NULL`
`javax.jdo.option.ConnectionURL`	A string de JDBC conexão para uma JDBC metastore.	`jdbc:derby:;databaseName=metastore_db;create=true`
`javax.jdo.option.ConnectionUserName`	O nome do usuário associado a um banco de dados do metastore.	`NULL`
`mapreduce.input.fileinputformat.split.maxsize`	O tamanho máximo de uma divisão durante o cálculo da divisão quando seu formato de entrada é`org.apache.hadoop.hive.ql.io.CombineHiveInputFormat` . O valor de 0 indica que não há limite.	0

Configuração	Descrição	Valor padrão
`tez.am.dag.cleanup.on.completion`	Opção que ativa a limpeza aleatória de dados quando concluída. DAG	`TRUE`
`tez.am.emr-serverless.launch.env.[KEY]`	Opção para definir a variável de *KEY* ambiente no processo Tez AM. Para Tez AM, esse valor substitui o valor. `hive.emr-serverless.launch.env.[KEY]`	
`tez.am.log.level`	O nível de registro raiz que o EMR Serverless passa para o mestre do aplicativo Tez.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMRO Serverless deve enviar ATS eventos após esse período após a solicitação de desligamento da manhã.	0
`tez.am.speculation.enabled`	Opção que causa o lançamento especulativo de tarefas mais lentas. Isso pode ajudar a reduzir a latência do trabalho quando algumas tarefas estão sendo executadas mais lentamente devido a máquinas defeituosas ou lentas. Compatível somente com o Amazon EMR 6.10.x e versões anteriores.	`FALSE`

Configuração	Descrição	Valor padrão
`tez.am.task.max.failed.attempts`	O número máximo de tentativas que podem falhar em uma tarefa específica antes que a tarefa falhe. Esse número não conta as tentativas encerradas manualmente.	3
`tez.am.vertex.cleanup.height`	Uma distância na qual, se todos os vértices dependentes estiverem completos, o Tez AM excluirá os dados de embaralhamento de vértices. Esse recurso é desativado quando o valor é 0. EMRAs versões 6.8.0 e posteriores da Amazon oferecem suporte a esse recurso.	0
`tez.client.asynchronous-stop`	Opção que faz com que o EMR Serverless envie ATS eventos antes de encerrar o driver do Hive.	`FALSE`
`tez.grouping.max-size`	O limite superior de tamanho (em bytes) de uma divisão agrupada. Esse limite evita divisões excessivamente grandes.	1073741824
`tez.grouping.min-size`	O limite de tamanho inferior (em bytes) de uma divisão agrupada. Esse limite evita muitas pequenas divisões.	16777216

Configuração	Descrição	Valor padrão
`tez.runtime.io.sort.mb`	O tamanho do buffer flexível quando o Tez classifica a saída é classificado.	O valor ideal é calculado com base na memória de tarefas Tez
`tez.runtime.unordered.output.buffer.size-mb`	O tamanho do buffer a ser usado se o Tez não gravar diretamente no disco.	O valor ideal é calculado com base na memória de tarefas Tez
`tez.shuffle-vertex-manager.max-src-fraction`	A fração das tarefas de origem que devem ser concluídas antes que o EMR Serverless programe todas as tarefas para o vértice atual (no caso de uma conexão). `ScatterGather` O número de tarefas prontas para agendamento no vértice atual é escalonado linearmente entre e. `min-fraction` `max-fraction` Isso padroniza o valor padrão out`ez.shuffle-vertex-manager.min-src-fraction` , o que for maior.	0.75
`tez.shuffle-vertex-manager.min-src-fraction`	A fração das tarefas de origem que devem ser concluídas antes que o EMR Serverless agende tarefas para o vértice atual (no caso de uma conexão). `ScatterGather`	0.25

Configuração	Descrição	Valor padrão
`tez.task.emr-serverless.launch.env.[ KEY]`	Opção para definir a variável de *KEY* ambiente no processo da tarefa Tez. Para tarefas Tez, esse valor substitui o valor. `hive.emr-serverless.launch.env.[ KEY]`	
`tez.task.log.level`	O nível de registro raiz que o EMR Serverless passa para as tarefas do Tez.	`INFO`
`tez.yarn.ats.event.flush.timeout.millis`	A quantidade máxima de tempo que o AM deve esperar até que os eventos sejam encerrados antes de serem encerrados.	300000

Exemplos de empregos no Hive

O exemplo de código a seguir mostra como executar uma consulta do Hive com o. `StartJobRun` API

```
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"
            }
        }]
    }'
```

Você pode encontrar exemplos adicionais de como executar trabalhos do Hive no repositório EMRServerless Samples. GitHub

EMRResiliência de trabalho sem servidor

EMRAs versões 7.1.0 e posteriores do Serverless incluem suporte à resiliência do trabalho, portanto, ele repete automaticamente qualquer trabalho com falha sem sua intervenção manual. Outro benefício da resiliência do trabalho é que o EMR Serverless transfere as execuções de tarefas para uma zona de disponibilidade (AZ) diferente, caso uma AZ tenha algum problema.

Para ativar a resiliência de um trabalho, defina a política de repetição para seu trabalho. Uma política de repetição garante que o EMR Serverless reinicie automaticamente um trabalho se ele falhar em algum momento. As políticas de repetição são compatíveis com trabalhos em lote e de streaming, para que você possa personalizar a resiliência do trabalho de acordo com seu caso de uso. A tabela a seguir compara os comportamentos e as diferenças da resiliência do trabalho em trabalhos em lote e de streaming.

	Trabalhos em lote	Trabalhos de streaming
Comportamento padrão	Não executa novamente o trabalho.	Sempre tente executar o trabalho novamente, pois o aplicativo cria pontos de verificação durante a execução do trabalho.
Ponto de nova tentativa	Os trabalhos em lote não têm pontos de verificação, então	Os trabalhos de streaming oferecem suporte a pontos

	Trabalhos em lote	Trabalhos de streaming
	o EMR Serverless sempre executa novamente o trabalho desde o início.	de verificação, para que você possa configurar a consulta de streaming para salvar o estado do tempo de execução e o progresso em um ponto de verificação no Amazon S3. EMRO Serverless retoma a execução do trabalho a partir do ponto de verificaç ão. Para obter mais informaçõ es, consulte Recuperação de falhas com o Checkpoint na documentação do Apache Spark.
Máximo de tentativas de repetição	Permite no máximo 10 tentativas.	Os trabalhos de streaming têm controle integrado de prevenção de thrash, então o aplicativo para de repetir os trabalhos se eles continuar em falhando após uma hora. O número padrão de novas tentativas em uma hora é de cinco tentativas. Você pode configurar esse número de novas tentativas entre 1 e 10. Você não pode personalizar o número máximo de tentativas. Um valor de 1 indica que não há novas tentativas.

Quando o EMR Serverless tenta executar novamente um trabalho, ele também indexa o trabalho com um número de tentativas, para que você possa acompanhar o ciclo de vida de um trabalho em todas as tentativas.

Você pode usar as API operações EMR sem servidor ou o AWS CLI para mudar a resiliência do trabalho ou ver informações relacionadas à resiliência do trabalho. Para obter mais informações, consulte o guia EMRServerless API.

Por padrão, o EMR Serverless não executa novamente trabalhos em lote. Para habilitar novas tentativas para trabalhos em lotes, configure o `maxAttempts` parâmetro ao iniciar a execução de um trabalho em lotes. O `maxAttempts` parâmetro é aplicável somente a trabalhos em lotes. O padrão é 1, o que significa não executar o trabalho novamente. Os valores aceitos são de 1 a 10, inclusive.

O exemplo a seguir demonstra como especificar um número máximo de 10 tentativas ao iniciar a execução de um trabalho.

```
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"
     }
}'
```

EMRO Serverless repete indefinidamente os trabalhos de streaming se eles falharem. Para evitar problemas devido a falhas irrecuperáveis repetidas, use o `maxFailedAttemptsPerHour` para configurar o controle de prevenção de problemas para novas tentativas de tarefas de streaming. Esse parâmetro permite especificar o número máximo de tentativas malsucedidas permitidas até uma hora antes que o EMR Serverless pare de tentar novamente. O padrão é cinco. Os valores aceitos são de 1 a 10, inclusive.

```
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"
     }
}'
```

Você também pode usar as outras API operações de execução de trabalhos para obter informações sobre trabalhos. Por exemplo, você pode usar o `attempt` parâmetro com a `GetJobRun` operação para obter detalhes sobre uma tentativa de trabalho específica. Se você não incluir o `attempt` parâmetro, a operação retornará informações sobre a última tentativa.

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

A `ListJobRunAttempts` operação retorna informações sobre todas as tentativas relacionadas à execução de um trabalho.

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

A `GetDashboardForJobRun` operação cria e retorna um URL que você pode usar para acessar o aplicativo UIs para a execução de um trabalho. O `attempt` parâmetro permite que você obtenha um URL para uma tentativa específica. Se você não incluir o `attempt` parâmetro, a operação retornará informações sobre a última tentativa.

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

Monitoramento de um trabalho com uma política de repetição

O suporte à resiliência de tarefas também adiciona o novo evento EMRServerless job run retry. EMRO Serverless publica esse evento em cada nova tentativa do trabalho. Você pode usar essa notificação para rastrear novas tentativas do trabalho. Para obter mais informações sobre eventos, consulte EventBridge Eventos da Amazon.

Registro com política de repetição

Toda vez que o EMR Serverless tenta novamente um trabalho, a tentativa gera seu próprio conjunto de registros. Para garantir que o EMR Serverless possa entregar com sucesso esses registros para o Amazon S3 e a CloudWatch Amazon sem sobrescrever EMR nenhum, o Serverless adiciona um prefixo ao formato do CloudWatch caminho de log do S3 e do nome do stream de log para incluir o número da tentativa do trabalho.

Veja a seguir um exemplo da aparência do formato.

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

Esse formato garante que o EMR Serverless publique todos os registros de cada tentativa do trabalho em seu próprio local designado no Amazon S3 e. CloudWatch Para obter mais detalhes, consulte Armazenamento de registros.

> **Note**
>
> EMRO Serverless usa esse formato de prefixo somente com todos os trabalhos de streaming e todos os trabalhos em lote que tenham a repetição ativada.

Configuração do Metastore

Um metastore do Hive é um local centralizado que armazena informações estruturais sobre suas tabelas, incluindo esquemas, nomes de partições e tipos de dados. Com o EMR Serverless, você pode manter esses metadados da tabela em um metastore que tenha acesso aos seus trabalhos.

Você tem duas opções para uma metastore do Hive:

- A ferramenta AWS Glue Data Catalog

- Uma metastore externa do Apache Hive

Usar o AWS Glue Data Catalog como metastore

Você pode configurar suas tarefas do Spark e do Hive para usar o AWS Glue Data Catalog como seu metastore. Recomendamos essa configuração quando você precisa de um metastore persistente ou compartilhado por diferentes aplicativos, serviços ou Contas da AWS. Para obter mais informações sobre o catálogo de dados, consulte Preenchendo o AWS Catálogo de dados Glue. Para obter mais informações sobre AWS Preços do Glue, consulte AWS Preços do Glue.

Você pode configurar seu trabalho EMR sem servidor para usar o AWS Glue Data Catalog ou no mesmo Conta da AWS como seu aplicativo ou em um diferente Conta da AWS.

Configurar o AWS Glue Data Catalog

Para configurar o Catálogo de Dados, escolha o tipo de aplicativo EMR sem servidor que você deseja usar.

Spark

Quando você usa o EMR Studio para executar seus trabalhos com aplicativos Spark EMR sem servidor, o AWS O Glue Data Catalog é o metastore padrão.

Quando você usa SDKs ou AWS CLI, você pode definir a `spark.hadoop.hive.metastore.client.factory.class` configuração `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` nos `sparkSubmit` parâmetros da execução do seu trabalho. O exemplo a seguir mostra como configurar o Catálogo de Dados com o AWS CLI.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-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"
        }
```

```
    }'
```

Como alternativa, você pode definir essa configuração ao criar uma nova `SparkSession` no seu código do 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

Para aplicativos EMR Serverless Hive, o Catálogo de Dados é o metastore padrão. Ou seja, quando você executa trabalhos em um aplicativo EMR Serverless Hive, o Hive registra as informações do metastore no Catálogo de Dados no mesmo Conta da AWS como seu aplicativo. Você não precisa de uma nuvem privada virtual (VPC) para usar o Catálogo de Dados como seu metastore.

Para acessar as tabelas do metastore do Hive, adicione o necessário AWS Políticas do Glue descritas em Configurando IAM permissões para AWS Glue.

Configure o acesso entre contas para EMR Serverless e AWS Glue Data Catalog

Para configurar o acesso entre contas para EMR Serverless, você deve primeiro fazer login no seguinte Contas da AWS:

- `AccountA`— Um Conta da AWS onde você criou um aplicativo EMR sem servidor.

- `AccountB`— Um Conta da AWS que contém um AWS Glue o catálogo de dados que você deseja que sua tarefa EMR sem servidor acesse.

1. Certifique-se de que um administrador ou outra identidade autorizada `AccountB` anexe uma política de recursos ao Catálogo de Dados em`AccountB`. Essa política concede permissões `AccountA` específicas entre contas para realizar operações em recursos no `AccountB` catálogo.

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::accountA:role/job-runtime-role-A"
        ]},
    "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:region:AccountB:catalog"]
  } ]
}
```

2. Adicione uma IAM política à função de tempo de execução do trabalho EMR sem servidor `AccountA` para que essa função possa acessar os recursos do Catálogo de Dados em. `AccountB`

```
{
  "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:region:AccountB:catalog"]
    }
  ]
}
```

3. Comece sua execução de trabalhos. Essa etapa é um pouco diferente dependendo `AccountA` do tipo de aplicativo EMR sem servidor.

   Spark

   Defina a `spark.hadoop.hive.metastore.glue.catalogid` propriedade na `hive-site` classificação conforme mostrado no exemplo a seguir. Substituir *ID do catálogo da conta* com o ID do catálogo de dados em`AccountB`.

   ```
   aws emr-serverless start-job-run \
   --application-id "application-id" \
   --execution-role-arn "job-role-arn" \
   --job-driver '{
       "sparkSubmit": {
           "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
           "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/
   hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
   hive/warehouse"
       }
     }' \
   ```

```
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "spark.hadoop.hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

Hive

Defina a `hive.metastore.glue.catalogid` propriedade na `hive-site` classificação conforme mostrado no exemplo a seguir. Substituir *ID do catálogo da conta* com o ID do catálogo de dados em`AccountB`.

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

Considerações ao usar o AWS Glue Data Catalog

Você pode adicionar auxiliares JARs `ADD JAR` em seus scripts do Hive. Para considerações adicionais, consulte Considerações ao usar AWS Catálogo de dados Glue.

Usando uma metastore externa do Hive

Você pode configurar suas tarefas EMR Serverless Spark e Hive para se conectarem a um metastore externo do Hive, como Amazon Aurora ou Amazon for My. RDS SQL Esta seção descreve como configurar um metastore Amazon RDS Hive, configurar seus trabalhos sem servidor e configurar seus VPC trabalhos EMR sem servidor para usar um metastore externo.

Crie uma metastore externa do Hive

1. Crie uma Amazon Virtual Private Cloud (AmazonVPC) com sub-redes privadas seguindo as instruções em Criar uma. VPC
2. Crie seu aplicativo EMR Serverless com sua nova Amazon VPC e sub-redes privadas. Quando você configura seu aplicativo EMR Serverless com umVPC, ele primeiro provisiona uma interface de rede elástica para cada sub-rede especificada. Em seguida, ele anexa o grupo de segurança especificado a essa interface de rede. Isso dá controle de acesso ao seu aplicativo. Para obter mais detalhes sobre como configurar seuVPC, consulteConfigurando o acesso VPC.
3. Crie um SQL banco de dados My SQL ou Aurora Postgre em uma sub-rede privada na sua Amazon. VPC Para obter informações sobre como criar um RDS banco de dados da Amazon, consulte Criação de uma RDS instância de banco de dados da Amazon.
4. Modifique o grupo de segurança do seu banco de dados My SQL ou Aurora para permitir JDBC conexões do seu grupo de segurança EMR sem servidor seguindo as etapas em Modificar uma instância de banco de dados Amazon. RDS Adicione uma regra para tráfego de entrada para o grupo de RDS segurança de um dos seus grupos de segurança EMR sem servidor.

Tipo	Protocolo	Intervalo de portas	Origem
Tudo TCP	TCP	3306	`emr-serverless-security-group`

Configurar as opções do Spark

Usando JDBC

Para configurar seu aplicativo EMR Serverless Spark para se conectar a um metastore Hive baseado em uma instância Amazon for RDS My ou SQL Amazon Aurora My, use uma conexão. SQL JDBC

Passe o `mariadb-connector-java.jar` with `--jars` nos `spark-submit` parâmetros da execução do seu trabalho.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

O exemplo de código a seguir é um script de ponto de entrada do Spark que interage com uma metastore do Hive na 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()
```

Usando o serviço de economia

Você pode configurar seu aplicativo EMR Serverless Hive para se conectar a um metastore do Hive com base em uma instância Amazon for RDS My ou SQL Amazon Aurora My. SQL Para fazer isso, execute um servidor econômico no nó principal de um EMR cluster existente da Amazon. Essa opção é ideal se você já tem um EMR cluster da Amazon com um servidor econômico que deseja usar para simplificar suas configurações de trabalho EMR sem servidor.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/thriftscript.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

O exemplo de código a seguir é um script de ponto de entrada (`thriftscript.py`) que usa o protocolo Thrift para se conectar a uma metastore do Hive. Observe que a `hive.metastore.uris` propriedade precisa ser configurada para ser lida em um metastore externo do Hive.

```
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()
```

Configurar as opções do Hive

Usando JDBC

Se você quiser especificar a localização de um banco de dados externo do Hive em uma instância do Amazon RDS My SQL ou do Amazon Aurora, você pode substituir a configuração padrão do metastore.

> **Note**
>
> No Hive, você pode realizar várias gravações em tabelas do metastore ao mesmo tempo. Se você compartilhar informações de metastore entre dois trabalhos, certifique-se de não gravar na mesma tabela de metastore simultaneamente, a menos que grave em partições diferentes da mesma tabela de metastore.

Defina as seguintes configurações na `hive-site` classificação para ativar o metastore externo do Hive.

```
{
    "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"
    }
}
```

Usando um servidor econômico

Você pode configurar seu aplicativo EMR Serverless Hive para se conectar a um metastore do Hive baseado em um Amazon for RDS My ou SQL Amazon Aurora M. ySQLinstance Para fazer isso, execute um servidor econômico no nó principal de um EMR cluster existente da Amazon. Essa opção é ideal se você já tem um EMR cluster da Amazon que executa um servidor econômico e deseja usar suas configurações de trabalho EMR sem servidor.

Defina as seguintes configurações na `hive-site` classificação para que o EMR Serverless possa acessar o metastore remoto do Thrift. Observe que você deve definir a `hive.metastore.uris` propriedade para ser lida em um metastore externo do Hive.

```
{
    "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"
    }
}
```

Considerações ao usar um metastore externo

- Você pode configurar bancos de dados compatíveis com o MariaDB como sua JDBC metastore. Exemplos desses bancos de dados são RDS para MariaDB, SQL My e Amazon Aurora.
- As metástores não são inicializadas automaticamente. Se sua metastore não for inicializada com um esquema para sua versão do Hive, use a Hive Schema Tool.

- EMRO Serverless não oferece suporte à autenticação Kerberos. Você não pode usar um servidor de metastore econômico com autenticação Kerberos com tarefas EMR Serverless Spark ou Hive.

Acessando dados do S3 em outro AWS conta da EMR Serverless

Você pode executar trabalhos do Amazon EMR Serverless a partir de um AWS conta e configure-os para acessar dados em buckets do Amazon S3 que pertencem a outra AWS conta. Esta página descreve como configurar o acesso entre contas ao S3 a partir do EMR Serverless.

Os trabalhos executados no EMR Serverless podem usar uma política de bucket do S3 ou uma função assumida para acessar dados no Amazon S3 a partir de uma outra AWS conta.

Pré-requisitos

Para configurar o acesso entre contas para o Amazon EMR Serverless, você deve concluir as tarefas enquanto estiver conectado a duas AWS contas:

- **`AccountA`**— Este é o AWS conta na qual você criou um aplicativo Amazon EMR Serverless. Antes de configurar o acesso entre contas, você deve ter o seguinte pronto nessa conta:
  - Um aplicativo Amazon EMR Serverless no qual você deseja executar trabalhos.
  - Uma função de execução de tarefas que tem as permissões necessárias para executar tarefas no aplicativo. Para obter mais informações, consulte Funções de tempo de execução de trabalho para Amazon EMR Serverless.
- **`AccountB`**— Este é o AWS conta que contém o bucket do S3 que você deseja que seus trabalhos do Amazon EMR Serverless acessem.

Use uma política de bucket do S3 para acessar dados do S3 entre contas

Para acessar o bucket do S3 em account B from account A, anexe a seguinte política ao bucket do S3 em account B.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions 1",
         "Effect": "Allow",
         "Principal": {
```

```
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB"
         ]
      },
      {
         "Sid": "Example permissions 2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB/*"
         ]
      }
   ]
}
```

Para obter mais informações sobre o acesso entre contas do S3 com políticas de bucket do S3, consulte Exemplo 2: Proprietário do bucket concedendo permissões de bucket entre contas no Guia do usuário do Amazon Simple Storage Service.

Use uma função assumida para acessar dados do S3 de várias contas

Outra forma de configurar o acesso entre contas para o Amazon EMR Serverless é com a ação do `AssumeRole` AWS Security Token Service (AWS STS). AWS STS é um serviço web global que permite solicitar credenciais temporárias com privilégios limitados para usuários. Você pode fazer API chamadas para o EMR Serverless e o Amazon S3 com as credenciais de segurança temporárias com as quais você cria. `AssumeRole`

As etapas a seguir ilustram como usar uma função assumida para acessar dados do S3 entre contas a partir do Serverless: EMR

1. Crie um bucket Amazon S3, *cross-account-bucket*, em`AccountB`. Para obter mais informações, consulte Criar um bucket no Guia do usuário do Amazon Simple Storage Service. Se desejar ter acesso entre contas para o DynamoDB, você também pode criar uma tabela do DynamoDB na `AccountB`. Para obter mais informações, consulte Criação de uma tabela do DynamoDB no Amazon DynamoDB Developer Guide.
2. Crie uma `Cross-Account-Role-B` IAM função `AccountB` que possa acessar o *cross-account-bucket*.
   a. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
   b. Escolha Perfis e crie um novo perfil: `Cross-Account-Role-B`. Para obter mais informações sobre como criar IAM funções, consulte Criação de IAM funções no Guia IAM do usuário.
   c. Crie uma IAM política que especifique as permissões `Cross-Account-Role-B` para acessar o *cross-account-bucket* Bucket S3, conforme demonstra a declaração de política a seguir. Em seguida, anexe a IAM política `Cross-Account-Role-B` a. Para obter mais informações, consulte Criação de IAM políticas no Guia IAM do usuário.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
            ]
        }
    ]
}
```

Se você precisar de acesso ao DynamoDB, crie IAM uma política que especifique as permissões para acessar a tabela do DynamoDB entre contas. Em seguida, anexe a IAM política `Cross-Account-Role-B` a. Para obter mais informações, consulte Amazon DynamoDB: Permite acesso a uma tabela específica no IAM Guia do usuário.

Veja a seguir uma política para permitir o acesso à tabela do DynamoDB`CrossAccountTable`.

```
{
```

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:MyRegion:AccountB:table/CrossAccountTable"
        }
    ]
}
```

3. Edite a relação de confiança para o perfil `Cross-Account-Role-B`.

   a. Para configurar a relação de confiança para a função, escolha a guia Relações de Confiança no IAM console para a função `Cross-Account-Role-B` que você criou na Etapa 2.

   b. Selecione Editar relação de confiança.

   c. Adicione o seguinte documento de política. Isso permite que `Job-Execution-Role-A` `AccountA` eu assuma o `Cross-Account-Role-B` papel.

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

4. Grant `Job-Execution-Role-A` `AccountA` no AWS STS `AssumeRole`permissão para assumir`Cross-Account-Role-B`.

   a. No IAM console para AWS conta`AccountA`, selecione`Job-Execution-Role-A`.

   b. Adicione a instrução de política a seguir ao `Job-Execution-Role-A` para permitir a ação `AssumeRole` no perfil `Cross-Account-Role-B`.

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

```
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
}
```

Exemplos de funções assumidas

Você pode usar uma única função assumida para acessar todos os recursos do S3 em uma conta ou, com o Amazon EMR 6.11 e superior, você pode configurar várias IAM funções a serem assumidas ao acessar diferentes buckets do S3 entre contas.

Tópicos

- Acesse os recursos do S3 com uma função assumida
- Acesse recursos do S3 com várias funções assumidas

Acesse os recursos do S3 com uma função assumida

> **Note**
>
> Quando você configura um trabalho para usar uma única função assumida, todos os recursos do S3 em todo o trabalho usam essa função, incluindo o `entryPoint` script.

Se você quiser usar uma única função assumida para acessar todos os recursos do S3 na conta B, especifique as seguintes configurações:

1. Especifique `fs.s3.customAWSCredentialsProvider` a EMRFS configuração para`spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRol`
2. Para o Spark, use `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` e especifique `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` as variáveis de ambiente no driver e nos executores.
3. Para o Hive `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, use e `tez.task.emr-`

`serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` para especificar as variáveis de ambiente no driver do Hive, no mestre do aplicativo Tez e nos contêineres de tarefas do Tez.

Os exemplos a seguir mostram como usar uma função assumida para iniciar uma execução de trabalho EMR sem servidor com acesso entre contas.

Spark

O exemplo a seguir mostra como usar uma função assumida para iniciar a execução de uma tarefa do EMR Serverless Spark com acesso entre contas ao 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":
 "spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
                "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

O exemplo a seguir mostra como usar uma função assumida para iniciar uma execução de trabalho do EMR Serverless Hive com acesso entre contas ao 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"
            }
        }]
    }'
```

Acesse recursos do S3 com várias funções assumidas

Com as versões 6.11.0 e posteriores do EMR Serverless, você pode configurar várias IAM funções a serem assumidas ao acessar diferentes buckets entre contas. Se você quiser acessar diferentes recursos do S3 com diferentes funções assumidas na conta B, use as seguintes configurações ao iniciar a execução do trabalho:

1. Especifique `fs.s3.customAWSCredentialsProvider` a EMRFS configuração para`com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCrede`
2. Especifique `fs.s3.bucketLevelAssumeRoleMapping` a EMRFS configuração para definir o mapeamento do nome do bucket do S3 para a IAM função na conta B a ser assumida. O valor deve estar no formato de`bucket1->role1;bucket2->role2`.

Por exemplo, você pode usar `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` para acessar o bucket `bucket1` e usar `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` para acessar o bucket`bucket2`. Os exemplos a seguir mostram como iniciar uma execução de trabalho EMR sem servidor com acesso entre contas por meio de várias funções assumidas.

Spark

O exemplo a seguir mostra como usar várias funções assumidas para criar uma execução de trabalho do EMR Serverless Spark.

```
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

Os exemplos a seguir mostram como usar várias funções assumidas para criar uma execução de trabalho do EMR Serverless Hive.

```
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"
            }
        }]
    }'
```

Solução de problemas no EMR Serverless

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon EMR Serverless.

Tópicos

- Erro: limite excedido para a capacidade máxima permitida.
- Erro: a capacidade máxima configurada foi excedida. Tente novamente mais tarde.
- Erro: o acesso ao S3 foi negado. Verifique as permissões de acesso ao S3 da função de tempo de execução do trabalho nos recursos necessários do S3.
- Erro ModuleNotFoundError: Nenhum módulo nomeado<module>. Consulte o guia do usuário sobre como usar bibliotecas python com o EMR Serverless.
- Erro: Não foi possível assumir a função de execução <role name>porque ela não existe ou não está configurada com a relação de confiança necessária.

Erro: limite excedido para a capacidade máxima permitida.

Esse erro indica que o EMR Serverless não conseguiu enviar o trabalho porque o aplicativo excedeu os limites máximos de capacidade configurados. Aumente os limites máximos de capacidade do aplicativo.

Erro: a capacidade máxima configurada foi excedida. Tente novamente mais tarde.

Esse erro indica que o EMR Serverless não pôde iniciar um novo trabalho porque o aplicativo excedeu os limites máximos de capacidade configurados. Aumente os limites máximos de capacidade do aplicativo.

Erro: o acesso ao S3 foi negado. Verifique as permissões de acesso ao S3 da função de tempo de execução do trabalho nos recursos necessários do S3.

Esse erro indica que seu trabalho não tem acesso aos recursos do S3. Verifique se a função de tempo de execução do trabalho tem permissão para acessar os recursos do S3 que o trabalho precisa usar. Para saber mais sobre as funções de tempo de execução, consulteFunções de tempo de execução de trabalho para Amazon EMR Serverless.

Erro ModuleNotFoundError: Nenhum módulo nomeado<module>. Consulte o guia do usuário sobre como usar bibliotecas python com o EMR Serverless.

Esse erro indica que um módulo Python não estava disponível para o trabalho do Spark. Verifique se as bibliotecas Python dependentes estão disponíveis para o trabalho. Para obter mais informações sobre como empacotar bibliotecas Python, consulte. Usando bibliotecas Python com Serverless EMR

Erro: Não foi possível assumir a função de execução <role name>porque ela não existe ou não está configurada com a relação de confiança necessária.

Esse erro indica que a função de tempo de execução do trabalho que você especificou para o trabalho não existe ou que a função não tem uma relação de confiança para permissões EMR sem

servidor. Para verificar se a IAM função existe e validar se você configurou a política de confiança da função corretamente, consulte as instruções emFunções de tempo de execução de trabalho para Amazon EMR Serverless.

Execute cargas de trabalho interativas com o EMR Serverless por meio do Studio EMR

Visão geral

Um aplicativo interativo é um aplicativo EMR sem servidor que tem recursos interativos habilitados. Com os aplicativos interativos Amazon EMR Serverless, você pode executar cargas de trabalho interativas com notebooks Jupyter que são gerenciados no Amazon Studio. EMR Isso ajuda engenheiros de dados, cientistas de dados e analistas de dados a usar o EMR Studio para executar análises interativas com conjuntos de dados em armazenamentos de dados como Amazon S3 e Amazon DynamoDB.

Os casos de uso de aplicativos interativos no EMR Serverless incluem o seguinte:

- Os engenheiros de dados usam a IDE experiência no EMR Studio para criar um ETL script. O script ingere dados locais, transforma os dados para análise e armazena os dados no Amazon S3.
- Os cientistas de dados usam notebooks para explorar conjuntos de dados e treinar modelos de aprendizado de máquina (ML) para detectar anomalias nos conjuntos de dados.
- Os analistas de dados exploram conjuntos de dados e criam scripts que geram relatórios diários para atualizar aplicativos, como painéis de negócios.

Pré-requisitos

Para usar cargas de trabalho interativas com o EMR Serverless, você deve atender aos seguintes requisitos:

- EMRAplicativos interativos sem servidor são compatíveis com o Amazon EMR 6.14.0 e versões posteriores.
- Para acessar seu aplicativo interativo, executar as cargas de trabalho que você envia e executar cadernos interativos do EMR Studio, você precisa de permissões e funções específicas. Para obter mais informações, consulte Permissões necessárias para cargas de trabalho interativas.

Permissões necessárias para cargas de trabalho interativas

Além das permissões básicas necessárias para acessar o EMR Serverless, você deve configurar permissões adicionais para sua IAM identidade ou função:

Para acessar seu aplicativo interativo

Configure as permissões de usuário e espaço de trabalho para o EMR Studio. Para obter mais informações, consulte Configurar permissões de usuário do EMR Studio no Amazon EMR Management Guide.

Para executar as cargas de trabalho que você envia com EMR o Serverless

Configure uma função de tempo de execução do trabalho. Para obter mais informações, consulte Crie uma função de tempo de execução do trabalho.

Para executar os notebooks interativos do Studio EMR

Adicione as seguintes permissões adicionais à IAM política para os usuários do Studio:

- **`emr-serverless:AccessInteractiveEndpoints`**- Concede permissão para acessar e se conectar ao aplicativo interativo que você especifica como`Resource`. Essa permissão é necessária para se conectar a um aplicativo EMR sem servidor a partir de um EMR Studio Workspace.
- **`iam:PassRole`**- Concede permissão para acessar a função de IAM execução que você planeja usar ao se conectar a um aplicativo. É necessária a `PassRole` permissão apropriada para se conectar a um aplicativo EMR sem servidor a partir de um EMR Studio Workspace.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessInteractiveEndpoints",
            "Resource": "arn:aws:emr-serverless:Region:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "interactive-execution-role-ARN",
```

```
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

Configurando aplicativos interativos

Use as seguintes etapas de alto nível para criar um aplicativo EMR sem servidor com recursos interativos do Amazon EMR Studio no AWS Management Console.

1. Siga as etapas Começando a usar o Amazon EMR Serverless para criar um aplicativo.
2. Em seguida, inicie um espaço de trabalho do EMR Studio e conecte-se a um aplicativo EMR sem servidor como opção de computação. Para obter mais informações, consulte a guia Carga de trabalho interativa na Etapa 2 da documentação EMRServerless Getting Started.

Quando você anexa um aplicativo a um Studio Workspace, o início do aplicativo é acionado automaticamente se ainda não estiver em execução. Você também pode pré-iniciar o aplicativo e mantê-lo pronto antes de anexá-lo ao Workspace.

Considerações sobre aplicativos interativos

- EMRAplicativos interativos sem servidor são compatíveis com o Amazon EMR 6.14.0 e versões posteriores.
- EMRO Studio é o único cliente integrado aos aplicativos interativos EMR sem servidor. Os seguintes recursos do EMR Studio não são compatíveis com aplicativos interativos EMR sem servidor: colaboração no espaço de trabalho, SQL Explorer e execução programática de notebooks.
- Aplicativos interativos são compatíveis apenas com o mecanismo Spark.
- Os aplicativos interativos oferecem suporte aos kernels Python 3 PySpark e Spark Scala.
- Você pode executar até 25 notebooks simultâneos em um único aplicativo interativo.
- Não há um endpoint ou API interface que suporte notebooks Jupyter auto-hospedados com aplicativos interativos.

- Para uma experiência de inicialização otimizada, recomendamos que você configure a capacidade pré-inicializada para drivers e executores e que você pré-inicie seu aplicativo. Ao pré-iniciar o aplicativo, você garante que ele esteja pronto quando quiser anexá-lo ao seu espaço de trabalho.

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

- Por padrão, `autoStopConfig` está habilitado para aplicativos. Isso desliga o aplicativo após 30 minutos de tempo ocioso. Você pode alterar essa configuração como parte `create-application` de sua `update-application` solicitação.
- Ao usar um aplicativo interativo, recomendamos que você configure uma capacidade pré-inicializada de kernels, drivers e executores para executar seus notebooks. Cada sessão interativa do Spark requer um kernel e um driver, então o EMR Serverless mantém um kernel worker pré-inicializado para cada driver pré-inicializado. Por padrão, o EMR Serverless mantém uma capacidade pré-inicializada de um kernel worker em todo o aplicativo, mesmo que você não especifique nenhuma capacidade pré-inicializada para drivers. Cada kernel worker usa 4 v CPU e 16 GB de memória. Para obter informações atuais sobre preços, consulte a página de EMRpreços da Amazon.
- Você deve ter uma cota CPU de serviço v suficiente em seu Conta da AWS para executar cargas de trabalho interativas. Se você não executa cargas de trabalho habilitadas para Lake Formation, recomendamos pelo menos 24 v. CPU Se você fizer isso, recomendamos pelo menos 28 CPU v.
- EMRO Serverless encerra automaticamente os kernels dos notebooks se eles ficarem inativos por mais de 60 minutos. EMRO Serverless calcula o tempo ocioso do kernel a partir da última atividade concluída durante a sessão do notebook. No momento, você não pode modificar a configuração de tempo limite de inatividade do kernel.
- Para habilitar o Lake Formation com cargas de trabalho interativas, `spark.emr-serverless.lakeformation.enabled` defina a configuração `true` abaixo da `spark-defaults` classificação no `runtime-configuration` objeto ao criar um aplicativo sem EMR servidor. Para saber mais sobre como habilitar o Lake Formation no EMR Serverless, consulte Habilitando o Lake Formation na Amazon. EMR

Execute cargas de trabalho interativas com o EMR Serverless por meio de um endpoint Apache Livy

Com as EMR versões 6.14.0 e superiores da Amazon, você pode criar e habilitar um endpoint Apache Livy enquanto cria um aplicativo EMR sem servidor e executa cargas de trabalho interativas por meio de seus notebooks auto-hospedados ou com um cliente personalizado. Um endpoint Apache Livy oferece os seguintes benefícios:

- Você pode se conectar com segurança a um endpoint Apache Livy por meio de notebooks Jupyter e gerenciar cargas de trabalho do Apache Spark com a interface do Apache Livy. REST
- Use as REST API operações do Apache Livy para aplicativos web interativos que usam dados das cargas de trabalho do Apache Spark.

Pré-requisitos

Para usar um endpoint Apache Livy com o EMR Serverless, você deve atender aos seguintes requisitos:

- Conclua as etapas em Introdução ao Amazon EMR Serverless.
- Para executar cargas de trabalho interativas por meio dos endpoints do Apache Livy, você precisa de certas permissões e funções. Para obter mais informações, consulte Permissões necessárias para cargas de trabalho interativas.

Permissões obrigatórias

Além das permissões necessárias para acessar o EMR Serverless, você também deve adicionar as seguintes permissões à sua IAM função para acessar um endpoint Apache Livy e executar aplicativos:

- `emr-serverless:AccessLivyEndpoints`— concede permissão para acessar e se conectar ao aplicativo habilitado para Livy que você especifica como. `Resource` Você precisa dessa permissão para executar as REST API operações disponíveis no endpoint Apache Livy.
- `iam:PassRole`— concede permissão para acessar a função de IAM execução ao criar a sessão do Apache Livy. EMRO Serverless usará essa função para executar suas cargas de trabalho.

- `emr-serverless:GetDashboardForJobRun`— concede permissão para gerar a interface do usuário do Spark Live e os links de registro do driver e fornece acesso aos registros como parte dos resultados da sessão do Apache Livy.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessLivyEndpoints",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "execution-role-ARN",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EMRServerlessDashboardAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:GetDashboardForJobRun",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        }
    ]
}
```

Conceitos básicos

1. Para criar um aplicativo compatível com o Apache Livy, execute o comando a seguir.

```
 aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
 --release-label <Amazon EMR-release-version>
--interactive-configuration '{"livyEndpointEnabled": true}'
```

2. Depois que o EMR Serverless criar seu aplicativo, inicie o aplicativo para disponibilizar o endpoint Apache Livy.

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

Use o comando a seguir para verificar o status do seu aplicativo. Quando o status chegar`STARTED`, você poderá acessar o endpoint Apache Livy.

```
aws emr-serverless get-application \
--region <AWS_REGION> --application-id >application_id>
```

3. Use o seguinte URL para acessar o endpoint:

```
https://_<application-id>_.livy.emr-serverless-
services._<AWS_REGION>_.amazonaws.com
```

Quando o endpoint estiver pronto, você poderá enviar cargas de trabalho com base no seu caso de uso. Você deve assinar todas as solicitações no endpoint com o SIGv4 protocolo e passar um cabeçalho de autorização. Você pode usar os seguintes métodos para executar cargas de trabalho:

- HTTPcliente — você deve enviar suas API operações de endpoint Apache Livy com um cliente personalizado. HTTP
- Kernel Sparkmagic — você deve executar localmente o kernel sparkmagic e enviar consultas interativas com os notebooks Jupyter.

HTTPclientes

Para criar uma sessão do Apache Livy, você deve enviar `emr-serverless.session.executionRoleArn` no `conf` parâmetro do corpo da solicitação. O exemplo a seguir é um exemplo de `POST /sessions` solicitação.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "<executionRoleArn>"
    }
```

```
}
```

A tabela a seguir descreve todas as operações disponíveis do Apache Livy. API

APIoperação	Descrição
GET/sessões	Retorna uma lista de todas as sessões interativ as ativas.
POST/sessões	Cria uma nova sessão interativa via spark ou pyspark.
GET/sessões/ <*sessionId* >	Retorna as informações da sessão.
GET/sessões/ <*sessionId* >/estado	Retorna o estado da sessão.
DELETE/sessões/ <*sessionId* >	Interrompe e exclui a sessão.
GET/sessões/ <*sessionId* >/declarações	Retorna todas as declarações em uma sessão.
POST/sessões/ <*sessionId* >/declarações	Executa uma declaração em uma sessão.
GET/sessões/ <*sessionId* >/declarações/ <*statementId* >	Retorna os detalhes da declaração especific ada em uma sessão.
POST/sessões/ <*sessionId* >/declarações/ <*statementId* >/cancelar	Cancela a declaração especificada nesta sessão.

Enviando solicitações para o endpoint Apache Livy

Você também pode enviar solicitações diretamente para o endpoint Apache Livy a partir de um cliente. HTTP Isso permite que você execute remotamente o código para seus casos de uso fora de um notebook.

Antes de começar a enviar solicitações para o endpoint, verifique se você instalou as seguintes bibliotecas:

```
pip3 install botocore awscrt requests
```

Veja a seguir um exemplo de script Python para enviar HTTP solicitações diretamente para um endpoint:

```
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())
```

Kernel Sparkmagic

Antes de instalar o sparkmagic, verifique se você configurou AWS credenciais na instância em que você deseja instalar o sparkmagic

1. Instale o sparkmagic seguindo as etapas de instalação. Observe que você só precisa executar as quatro primeiras etapas.
2. O kernel sparkmagic suporta autenticadores personalizados, para que você possa integrar um autenticador ao kernel sparkmagic para que cada solicitação seja assinada. SIGv4
3. Instale o EMR autenticador personalizado sem servidor.

```
pip install emr-serverless-customauth
```

4. Agora, forneça o caminho para o autenticador personalizado e o endpoint Apache Livy URL no arquivo json de configuração sparkmagic. Use o comando a seguir para abrir o arquivo de configuração.

```
vim ~/.sparkmagic/config.json
```

Veja a seguir um exemplo de arquivo `config.json`.

```
{
"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. Inicie o laboratório Jupyter. Ele deve usar a autenticação personalizada que você configurou na última etapa.
6. Em seguida, você pode executar os seguintes comandos do notebook e seu código para começar.

```
%%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
```

Internamente, cada instrução chama cada uma das API operações do Apache Livy por meio do endpoint configurado do Apache Livy. URL Em seguida, você pode escrever suas instruções de acordo com seu caso de uso.

Considerações

Considere as seguintes considerações ao executar cargas de trabalho interativas por meio de endpoints do Apache Livy.

- EMRO Serverless mantém o isolamento no nível da sessão usando o chamador principal. O chamador principal que cria a sessão é o único que pode acessar essa sessão. Para um isolamento mais granular, você pode configurar uma identidade de origem ao assumir as credenciais. Nesse caso, o EMR Serverless impõe o isolamento em nível de sessão com base no chamador principal e na identidade de origem. Para obter mais informações sobre a identidade de origem, consulte Monitorar e controlar ações tomadas com funções assumidas.
- Os endpoints Apache Livy são compatíveis com as versões 6.14.0 e superiores do EMR Serverless.

- Os endpoints do Apache Livy são compatíveis somente com o mecanismo Apache Spark.
- Os endpoints Apache Livy são compatíveis com Scala Spark e. PySpark
- Por padrão, `autoStopConfig` está habilitado em seus aplicativos. Isso significa que os aplicativos são encerrados após 15 minutos de inatividade. Você pode alterar essa configuração como parte `create-application` de sua `update-application` solicitação.
- Você pode executar até 25 sessões simultâneas em um único aplicativo habilitado para endpoint Apache Livy.
- Para obter a melhor experiência de inicialização, recomendamos que você configure a capacidade pré-inicializada para drivers e executores.
- Você deve iniciar manualmente seu aplicativo antes de se conectar ao endpoint Apache Livy.
- Você deve ter uma cota CPU de serviço v suficiente em seu Conta da AWS para executar cargas de trabalho interativas com o endpoint Apache Livy. Recomendamos pelo menos 24 CPU v.
- O tempo limite padrão da sessão do Apache Livy é de 1 hora. Se você não executar instruções por uma hora, o Apache Livy excluirá a sessão e liberará o driver e os executores. Você não pode alterar essa configuração.
- Somente sessões ativas podem interagir com um endpoint Apache Livy. Depois que a sessão for concluída, cancelada ou encerrada, você não poderá acessá-la por meio do endpoint Apache Livy.

Logging e monitoramento

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho dos aplicativos e EMR trabalhos sem servidor. Você deve coletar dados de monitoramento de todas as partes de suas soluções EMR sem servidor para poder depurar com mais facilidade uma falha multiponto, caso ocorra.

Tópicos

Armazenando registros

Para monitorar o progresso do seu trabalho no EMR Serverless e solucionar falhas de trabalho, você pode escolher como o EMR Serverless armazena e veicula os registros do aplicativo. Ao enviar uma execução de trabalho, você pode especificar armazenamento gerenciado, Amazon S3 e Amazon CloudWatch como suas opções de registro.

Com CloudWatch, você pode especificar os tipos e locais de registro que deseja usar ou aceitar os tipos e locais padrão. Para obter mais informações sobre CloudWatch registros, consultethe section called “Amazon CloudWatch”. Com o armazenamento gerenciado e o registro do S3, a tabela a seguir mostra os locais de log e a disponibilidade da interface do usuário que você pode esperar se escolher armazenamento gerenciado, buckets do Amazon S3 ou ambos.

Opção	Registros de eventos	Logs de contêineres	UI do aplicativo
Armazenamento gerenciado	Armazenado em armazenamento gerenciado	Armazenado em armazenamento gerenciado	Compatível

Opção	Registros de eventos	Logs de contêineres	UI do aplicativo
Armazenamento gerenciado e bucket S3	Armazenado em ambos os lugares	Armazenado no bucket S3	Compatível
Bucket do Amazon S3	Armazenado no bucket S3	Armazenado no bucket S3	Não suportado [1]

[1] Recomendamos que você mantenha a opção Armazenamento gerenciado selecionada. Caso contrário, você não poderá usar o aplicativo integradoUIs.

Registro EMR sem servidor com armazenamento gerenciado

Por padrão, o EMR Serverless armazena registros de aplicativos com segurança no armazenamento gerenciado EMR da Amazon por no máximo 30 dias.

> **Note**
>
> Se você desativar a opção padrão, a Amazon não EMR poderá solucionar seus trabalhos em seu nome.

Para desativar essa opção do EMR Studio, desmarque a opção Permitir AWS para reter registros por 30 dias, marque a caixa de seleção na seção Configurações adicionais da página Enviar trabalho.

Para desativar essa opção a partir do AWS CLI, use a `managedPersistenceMonitoringConfiguration` configuração ao enviar uma execução de trabalho.

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }
}
```

Registro EMR sem servidor com buckets Amazon S3

Antes que seus trabalhos possam enviar dados de log para o Amazon S3, você deve incluir as seguintes permissões na política de permissões para a função de tempo de execução do trabalho. *DOC-EXAMPLE-BUCKET-LOGGING*Substitua pelo nome do seu bucket de registro.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        }
    ]
}
```

Para configurar um bucket do Amazon S3 para armazenar registros do AWS CLI, use a `s3MonitoringConfiguration` configuração ao iniciar a execução de um trabalho. Para fazer isso, forneça o seguinte `--configuration-overrides` na configuração.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/"
        }
    }
}
```

Para trabalhos em lotes que não têm novas tentativas ativadas, o EMR Serverless envia os registros para o seguinte caminho:

```
'/applications/<applicationId>/jobs/<jobId>'
```

EMRAs versões sem servidor 7.1.0 e posteriores oferecem suporte a tentativas de repetição para trabalhos de streaming e trabalhos em lote. Se você executar um trabalho com novas tentativas

ativadas, o EMR Serverless adicionará automaticamente um número de tentativas ao prefixo do caminho do registro, para que você possa distinguir e rastrear melhor os registros.

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

Registro EMR sem servidor com a Amazon CloudWatch

Ao enviar um trabalho para um aplicativo EMR sem servidor, você pode escolher a Amazon CloudWatch como uma opção para armazenar os registros do seu aplicativo. Isso permite que você use recursos de análise de CloudWatch registros, como CloudWatch Logs Insights e Live Tail. Você também pode transmitir registros CloudWatch para outros sistemas, por exemplo, OpenSearch para análise posterior.

EMRO Serverless fornece registro em tempo real para registros de drivers. Você pode visualizar os registros em tempo real com o recurso de cauda CloudWatch ao vivo ou por meio CloudWatch CLI de comandos finais.

Por padrão, o CloudWatch registro está desativado para EMR Serverless. Para habilitá-lo, consulte a configuração emAWS CLI.

> **Note**
>
> A Amazon CloudWatch publica registros em tempo real, portanto, gera mais recursos dos trabalhadores. Se você escolher uma baixa capacidade de trabalho, o impacto no tempo de execução do trabalho poderá aumentar. Se você habilitar o CloudWatch registro, recomendamos que escolha uma capacidade de trabalho maior. Também é possível que a publicação de registros seja reduzida se a taxa de transações por segundo (TPS) for muito baixa para. `PutLogEvents` A configuração de CloudWatch limitação é global para todos os serviços, incluindo EMR Serverless. Para obter mais informações, consulte Como determino a limitação em meus registros? CloudWatch em AWS re:post.

Permissões necessárias para fazer login com CloudWatch

Antes que seus trabalhos possam enviar dados de log para a Amazon CloudWatch, você deve incluir as seguintes permissões na política de permissões para a função de tempo de execução do trabalho.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:Região da AWS:111122223333:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:PutLogEvents",
               "logs:CreateLogGroup",
               "logs:CreateLogStream",
               "logs:DescribeLogStreams"
            ],
            "Resource": [
               "arn:aws:logs:Região da AWS:111122223333:log-group:my-log-group-name:*"
            ]
        }
    ]
}
```

AWS CLI

Para configurar a Amazon CloudWatch para armazenar registros para EMR Serverless a partir do AWS CLI, use a `cloudWatchLoggingConfiguration` configuração ao iniciar a execução de um trabalho. Para fazer isso, forneça as seguintes substituições de configuração. Opcionalmente, você também pode fornecer um nome de grupo de log, nome de prefixo de fluxo de log, tipos de log e uma chave de criptografia. ARN

Se você não especificar os valores opcionais, CloudWatch publicará os registros em um grupo de registros padrão`/aws/emr-serverless`, com o fluxo `/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`* de registros padrão.

EMRAs versões sem servidor 7.1.0 e posteriores oferecem suporte a tentativas de repetição para trabalhos de streaming e trabalhos em lote. Se você habilitou novas tentativas para um trabalho, o EMR Serverless adiciona automaticamente um número de tentativas ao prefixo do caminho do registro, para que você possa distinguir e rastrear melhor os registros.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/worker-type'
```

O seguinte mostra a configuração mínima necessária para ativar o CloudWatch registro de registros na Amazon com as configurações padrão para EMR Serverless:

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true
         }
     }
}
```

O exemplo a seguir mostra todas as configurações obrigatórias e opcionais que você pode especificar ao ativar o Amazon CloudWatch Logging para EMR Serverless. Os `logTypes` valores suportados também estão listados abaixo deste exemplo.

```
{
    "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
             }
         }
     }
}
```

Por padrão, o EMR Serverless publica somente os registros do driver stdout e stderr em. CloudWatch Se você quiser outros registros, poderá especificar uma função de contêiner e os tipos de registro correspondentes com o `logTypes` campo.

A lista a seguir mostra os tipos de trabalhadores compatíveis que você pode especificar para a `logTypes` configuração:

Spark

- `SPARK_DRIVER : ["STDERR", "STDOUT"]`

- `SPARK_EXECUTOR : ["STDERR", "STDOUT"]`

Hive

- `HIVE_DRIVER : ["STDERR", "STDOUT", "HIVE_LOG", "TEZ_AM"]`
- `TEZ_TASK : ["STDERR", "STDOUT", "SYSTEM_LOGS"]`

Troncos rotativos

O Amazon EMR Serverless pode alternar registros de aplicativos e registros de eventos do Spark. A rotação de registros ajuda com o problema de trabalhos de longa execução, gerando grandes arquivos de log que podem ocupar todo o espaço em disco. A rotação de registros ajuda a economizar armazenamento em disco e reduz a quantidade de falhas de trabalho porque você não tem mais espaço no disco.

A rotação de registros é ativada por padrão e está disponível somente para trabalhos do Spark.

Registros de eventos do Spark

> **Note**
>
> A rotação do registro de eventos do Spark está disponível em todas as gravadoras de EMR lançamento da Amazon.

Em vez de gerar um único arquivo de registro de eventos, o EMR Serverless gira o registro de eventos em um intervalo de tempo regular e remove os arquivos de registro de eventos mais antigos. A rotação de registros não afeta os registros enviados para o bucket do S3.

Registros do aplicativo Spark

> **Note**
>
> A rotação do registro do aplicativo Spark está disponível em todas as gravadoras de EMR lançamento da Amazon.

EMRO Serverless também gira os registros do aplicativo Spark para drivers e executores, como arquivos e. `stdout` `stderr` Você pode acessar os arquivos de log mais recentes escolhendo os

links de log no Studio usando os links Spark History Server e Live UI. Os arquivos de log são as versões truncadas dos registros mais recentes. Para ver os registros rotacionados mais antigos, você deve especificar uma localização do Amazon S3 ao armazenar os registros. Consulte Logging for EMR Serverless with Amazon S3 buckets para obter mais informações.

Você pode encontrar os arquivos de log mais recentes no seguinte local. EMRO Serverless atualiza os arquivos a cada 15 segundos. Esses arquivos podem variar de 0 MB a 128 MB.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/stderr.gz
```

O local a seguir contém os arquivos rotacionados mais antigos. Cada arquivo tem 128 MB.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/archived/
stderr_<index>.gz
```

O mesmo comportamento também se aplica aos executores do Spark. Essa alteração é aplicável somente ao registro do S3. A rotação de registros não introduz nenhuma alteração nos fluxos de registros enviados para a Amazon CloudWatch.

EMRAs versões sem servidor 7.1.0 e posteriores oferecem suporte a tentativas de repetição para trabalhos de streaming e em lote. Se você habilitou novas tentativas com seu trabalho, o EMR Serverless adicionará um prefixo ao caminho de registro desses trabalhos para que você possa rastrear e distinguir melhor os registros uns dos outros. Esse caminho contém todos os registros girados.

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

Criptografando registros

Criptografando registros EMR sem servidor com armazenamento gerenciado

Para criptografar registros no armazenamento gerenciado com sua própria KMS chave, use a `managedPersistenceMonitoringConfiguration` configuração ao enviar uma execução de trabalho.

```
{
```

```
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration" : {
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Criptografando registros EMR sem servidor com buckets do Amazon S3

Para criptografar registros em seu bucket do Amazon S3 com sua KMS própria chave, use `s3MonitoringConfiguration` a configuração ao enviar uma execução de trabalho.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/",
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Criptografando registros EMR sem servidor com a Amazon CloudWatch

Para criptografar registros na Amazon CloudWatch com sua própria KMS chave, use a `cloudWatchLoggingConfiguration` configuração ao enviar uma execução de trabalho.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true,
            "encryptionKeyArn": "key-arn"
         }
     }
}
```

Permissões necessárias para criptografia de registros

Nesta seção

- Permissões de chave de criptografia para Amazon CloudWatch

Permissões de usuário necessárias

O usuário que envia o trabalho ou visualiza os registros ou o aplicativo UIs deve ter permissões para usar a chave. Você pode especificar as permissões na política de KMS chaves ou na IAM política do usuário, grupo ou função. Se o usuário que envia o trabalho não tiver as permissões KMS principais, o EMR Serverless rejeitará o envio da execução do trabalho.

Exemplo de política de chaves

A política de chaves a seguir fornece as permissões para `kms:GenerateDataKey` e `kms:Decrypt`:

```
{
    "Effect": "Allow",
    "Principal":{
       "AWS": "arn:aws:iam::111122223333:user/user-name"
     },
     "Action": [
       "kms:GenerateDataKey",
       "kms:Decrypt"
      ],
     "Resource": "*"
 }
```

Exemplo IAM de política

A IAM política a seguir fornece as permissões para `kms:GenerateDataKey` e `kms:Decrypt`:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Para iniciar a interface do usuário do Spark ou do Tez, você deve dar aos seus usuários, grupos ou funções permissões para acessar o `emr-serverless:GetDashboardForJobRun` API seguinte:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "emr-serverless:GetDashboardForJobRun"
      ]
   }
}
```

Permissões de chave de criptografia para Amazon S3 e armazenamento gerenciado

Ao criptografar registros com sua própria chave de criptografia no armazenamento gerenciado ou nos buckets do S3, você deve configurar as permissões da KMS chave da seguinte forma.

O `emr-serverless.amazonaws.com` diretor deve ter as seguintes permissões na política da KMS chave:

```
{
    "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"
       }
     }
 }
```

Como prática recomendada de segurança, recomendamos que você adicione uma chave de `aws:SourceArn` condição à política de KMS chaves. A chave de condição IAM global

`aws:SourceArn` ajuda a garantir que o EMR Serverless use a KMS chave somente para um aplicativo. ARN

A função de tempo de execução do trabalho deve ter as seguintes permissões em sua IAM política:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Permissões de chave de criptografia para Amazon CloudWatch

Para associar a KMS chave ARN ao seu grupo de registros, use a IAM política a seguir para a função de tempo de execução do trabalho.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:Região da AWS:111122223333:log-group:my-log-group-name:*"
        ]
    }
}
```

Configure a política de KMS chaves para conceder KMS permissões à Amazon CloudWatch:

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement":
```

```
    {
       "Effect": "Allow",
       "Principal": {
           "Service": "logs.Região da AWS.amazonaws.com"
       },
       "Action": [
           "kms:Decrypt",
           "kms:GenerateDataKey",
       ],
       "Resource": "*",
       "Condition": {
           "ArnLike": {
               "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:Região da
 AWS:111122223333:*"
           }
       }
    }
}
```

Configurar propriedades do Apache Log4j2 para Amazon Serverless EMR

Esta página descreve como configurar propriedades personalizadas do Apache Log4j 2.x para EMR trabalhos sem servidor em. `StartJobRun` Se você quiser configurar as classificações do Log4j no nível do aplicativo, consulte. Configuração padrão do aplicativo para EMR Serverless

Configurar propriedades do Spark Log4j2 para Amazon Serverless EMR

Com as EMR versões 6.8.0 e superiores da Amazon, você pode personalizar as propriedades do Apache Log4j 2.x para especificar configurações de log refinadas. Isso simplifica a solução de problemas de suas tarefas do Spark no EMR Serverless. Para configurar essas propriedades, use `spark-driver-log4j2` `spark-executor-log4j2` as classificações e.

Tópicos

- Classificações Log4j2 para Spark
- Exemplo de configuração do Log4j2 para o Spark
- Log4j2 em exemplos de trabalhos do Spark
- Considerações sobre o Log4j2 para o Spark

Classificações Log4j2 para Spark

Para personalizar as configurações de log do Spark, use as seguintes classificações com. `applicationConfiguration` Para configurar as propriedades do Log4j 2.x, use o seguinte. `properties`

`spark-driver-log4j2`

Essa classificação define os valores no `log4j2.properties` arquivo para o driver.

`spark-executor-log4j2`

Essa classificação define os valores no `log4j2.properties` arquivo para o executor.

Exemplo de configuração do Log4j2 para o Spark

O exemplo a seguir mostra como enviar uma tarefa do Spark para personalizar as configurações do Log4j2 `applicationConfiguration` para o driver e o executor do Spark.

Para configurar as classificações do Log4j no nível do aplicativo, em vez de quando você envia o trabalho, consulte. Configuração padrão do aplicativo para 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 em exemplos de trabalhos do Spark

Os exemplos de código a seguir demonstram como criar um aplicativo Spark enquanto você inicializa uma configuração personalizada do Log4j2 para o aplicativo.

Python

Example - Usando o Log4j2 para um trabalho do Spark com 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()
```

Para personalizar o Log4j2 para o driver ao executar uma tarefa do Spark, você pode usar a seguinte configuração:

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example - Usando o Log4j2 para um trabalho do Spark com 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()
  }
}
```

Para personalizar o Log4j2 para o driver ao executar uma tarefa do Spark, você pode usar a seguinte configuração:

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.ExampleClass.level": "info",
          "logger.ExampleClass.name": "ExampleClass"
      }
}
```

Considerações sobre o Log4j2 para o Spark

As seguintes propriedades do Log4j2.x não são configuráveis para processos do Spark:

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

Para obter informações detalhadas sobre as propriedades do Log4j2.x que você pode configurar, consulte o arquivo em. `log4j2.properties.template` GitHub

Monitoramento EMR sem servidor

Esta seção aborda as maneiras pelas quais você pode monitorar seus aplicativos e trabalhos do Amazon EMR Serverless.

Tópicos

- Monitoramento de aplicativos e EMR trabalhos sem servidor
- Monitore as métricas do Spark com o Amazon Managed Service para Prometheus
- EMRMétricas de uso sem servidor

Monitoramento de aplicativos e EMR trabalhos sem servidor

Com CloudWatch as métricas da Amazon para EMR Serverless, você pode receber CloudWatch métricas de 1 minuto e acessar CloudWatch painéis para visualizar as near-real-time operações e o desempenho de seus aplicativos sem servidor. EMR

EMRO Serverless envia métricas a CloudWatch cada minuto. EMRO Serverless emite essas métricas no nível do aplicativo, bem como no cargo, no tipo de trabalhador e nos níveis. capacity-allocation-type

Para começar, use o modelo de CloudWatch painel EMR Serverless fornecido no GitHub repositório EMR Serverless e implante-o.

> **Note**
>
> EMRAs cargas de trabalho interativas sem servidor têm apenas o monitoramento em nível de aplicativo ativado e têm uma nova dimensão de tipo de trabalhador,. `Spark_Kernel` Para monitorar e depurar suas cargas de trabalho interativas, você pode visualizar os registros e a interface do usuário do Apache Spark de dentro do seu Studio Workspace. EMR

A tabela abaixo descreve as dimensões EMR sem servidor disponíveis no namespace. `AWS/EMRServerless`

Dimensões para métricas EMR sem servidor

Dimensão	Descrição
`ApplicationId`	Filtros para todas as métricas de um EMR aplicativo sem servidor.
`JobId`	Filtros para todas as métricas da execução de uma tarefa EMR sem servidor.
`WorkerType`	Filtros para todas as métricas de um determina do tipo de trabalhador. Por exemplo, você pode filtrar

Dimensão	Descrição	
	por SPARK_DRIVER e SPARK_EXECUTORS para trabalhos do Spark.	
CapacityAllocationType	Filtros para todas as métricas de um determinado tipo de alocação de capacidade. Por exemplo, você pode filtrar PreInitCapacity pela capacidade pré-inicializada e OnDemandCapacity por todo o resto.	

Monitoramento em nível de aplicativo

Você pode monitorar o uso da capacidade no nível do aplicativo EMR Serverless com as métricas da Amazon CloudWatch . Você também pode configurar uma visualização única para monitorar o uso da capacidade do aplicativo em um CloudWatch painel.

EMRMétricas de aplicativos sem servidor

Métrica	Descrição	Dimensão primária	Dimensão secundária
CPUAllocated	Os números totais de vCPUs alocados.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
IdleWorkerCount	O número total de trabalhadores ociosos.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
MaxCPUAllowed	O máximo CPU permitido para o aplicativo.	ApplicationId	N/D

Métrica	Descrição	Dimensão primária	Dimensão secundária
MaxMemoryAllowed	A memória máxima em GB permitida para o aplicativo.	ApplicationId	N/D
MaxStorageAllowed	O armazenamento máximo em GB permitido para o aplicativo.	ApplicationId	N/D
MemoryAllocated	A memória total em GB alocada.	ApplicationId	ApplicationId, WorkerType, CapacityAllocationType
PendingCreationWorkerCount	O número total de trabalhadores pendentes de criação.	ApplicationId	ApplicationId, WorkerType, CapacityAllocationType
RunningWorkerCount	O número total de trabalhadores em uso pelo aplicativo.	ApplicationId	ApplicationId, WorkerType, CapacityAllocationType
StorageAllocated	O armazenamento total em disco em GB alocado.	ApplicationId	ApplicationId, WorkerType, CapacityAllocationType
TotalWorkerCount	O número total de trabalhadores disponíveis.	ApplicationId	ApplicationId, WorkerType, CapacityAllocationType

Monitoramento em nível de trabalho

O Amazon EMR Serverless envia as seguintes métricas de nível de trabalho para Amazon CloudWatch a cada um minuto. Você pode visualizar os valores métricos para execuções de tarefas agregadas por estado de execução de tarefas. A unidade para cada uma das métricas é contagem.

EMRMétricas de nível de trabalho sem servidor

Métrica	Descrição	Dimensão primária
`SubmittedJobs`	O número de trabalhos em um estado Enviado.	`ApplicationId`
`PendingJobs`	O número de trabalhos em um estado pendente.	`ApplicationId`
`ScheduledJobs`	O número de trabalhos em um estado programado.	`ApplicationId`
`RunningJobs`	O número de trabalhos em um estado em execução.	`ApplicationId`
`SuccessJobs`	O número de trabalhos em um estado de sucesso.	`ApplicationId`
`FailedJobs`	O número de trabalhos em um estado com falha.	`ApplicationId`
`CancellingJobs`	O número de trabalhos em um estado de cancelamento.	`ApplicationId`
`CancelledJobs`	O número de trabalhos em um estado Cancelado.	`ApplicationId`

Você pode monitorar métricas específicas do mecanismo para trabalhos EMR sem servidor em execução e concluídos com o aplicativo específico do mecanismo. UIs Ao visualizar a interface de usuário de um trabalho em execução, você vê a interface do usuário do aplicativo ativa com atualizações em tempo real. Ao visualizar a interface de usuário de um trabalho concluído, você vê a interface de usuário persistente do aplicativo.

Execução de trabalhos

Para suas tarefas EMR sem servidor em execução, você pode visualizar uma interface em tempo real que fornece métricas específicas do mecanismo. Você pode usar a interface do Apache Spark ou a interface do usuário do Hive Tez para monitorar e depurar seus trabalhos. Para acessá-losUIs, use o console do EMR Studio ou solicite um URL endpoint seguro com o AWS Command Line Interface.

Trabalhos concluídos

Para seus trabalhos concluídos EMR sem servidor, você pode usar o Spark History Server ou a interface do usuário persistente do Hive Tez para visualizar detalhes, estágios, tarefas e métricas das execuções de trabalhos do Spark ou do Hive. Para acessá-losUIs, use o console do EMR Studio ou solicite um URL endpoint seguro com o AWS Command Line Interface.

Monitoramento em nível de funcionário

O Amazon EMR Serverless envia as seguintes métricas de nível de funcionário que estão disponíveis no `AWS/EMRServerless` namespace e no `Job Worker Metrics` grupo de métricas para a Amazon. CloudWatch EMRO Serverless coleta pontos de dados de trabalhadores individuais durante a execução do trabalho no nível do trabalho, no tipo de trabalhador e no nível. capacity-allocation-type Você pode usar `ApplicationId` como uma dimensão para monitorar vários trabalhos que pertencem ao mesmo aplicativo.

EMRMétricas sem servidor em nível de funcionário

Métrica	Descrição	Unidade	Dimensão primária	Dimensão secundária
`WorkerCpuAllocated`	O número total de v CPU núcleos alocados para trabalhadores em uma execução de trabalho.	Nenhum	`JobId`	`ApplicationId` , `WorkerType` , e `CapacityAllocationType`
`WorkerCpuUsed`	O número total de CPU núcleos	Nenhum	`JobId`	`ApplicationId` ,

Métrica	Descrição	Unidade	Dimensão primária	Dimensão secundária
	v utilizados pelos trabalhadores em uma execução de trabalho.			`WorkerType` , e `CapacityAllocationType`
`WorkerMemoryAllocated`	A memória total em GB alocada para trabalhadores em uma execução de trabalho.	Gigabytes (GB)	`JobId`	`ApplicationId` , `WorkerType` , e `CapacityAllocationType`
`WorkerMemoryUsed`	A memória total em GB utilizada pelos trabalhadores em uma execução de trabalho.	Gigabytes (GB)	`JobId`	`ApplicationId` , `WorkerType` , e `CapacityAllocationType`
`WorkerEphemeralStorageAllocated`	O número de bytes de armazenamento temporário alocados para trabalhadores em uma execução de trabalho.	Gigabytes (GB)	`JobId`	`ApplicationId` , `WorkerType` , e `CapacityAllocationType`

Métrica	Descrição	Unidade	Dimensão primária	Dimensão secundária
`WorkerEphemeralStorageUsed`	O número de bytes de armazenamento temporário usados pelos trabalhadores em uma execução de trabalho.	Gigabytes (GB)	`JobId`	`ApplicationId`, `WorkerType`, e `CapacityAllocationType`
`WorkerStorageReadBytes`	O número de bytes lidos do armazenamento pelos trabalhadores em uma execução de trabalho.	Bytes	`JobId`	`ApplicationId`, `WorkerType`, e `CapacityAllocationType`
`WorkerStorageWriteBytes`	O número de bytes gravados no armazenamento pelos trabalhadores em uma execução de trabalho.	Bytes	`JobId`	`ApplicationId`, `WorkerType`, e `CapacityAllocationType`

As etapas abaixo descrevem como visualizar os vários tipos de métricas.

Console

Para acessar a interface do usuário do seu aplicativo com o console

1. Navegue até seu aplicativo EMR Serverless no EMR Studio com as instruções em Introdução ao console.
2. Para visualizar aplicativos UIs e registros específicos do mecanismo para um trabalho em execução:
   a. Escolha um emprego com um `RUNNING` status.
   b. Selecione o trabalho na página de detalhes do aplicativo ou navegue até a página de detalhes do trabalho do seu trabalho.
   c. No menu suspenso Display UI, escolha Spark UI ou Hive Tez UI para navegar até a interface do aplicativo para seu tipo de trabalho.
   d. Para ver os registros do mecanismo Spark, navegue até a guia Executors na interface do Spark e escolha o link Logs do driver. Para visualizar os registros do mecanismo do Hive, escolha o link Registros apropriado DAG na interface do usuário do Hive Tez.
3. Para visualizar a aplicação UIs e os registros específicos do motor de um trabalho concluído:
   a. Escolha um emprego com um `SUCCESS` status.
   b. Selecione a vaga na página de detalhes da candidatura ou navegue até a página de detalhes da vaga.
   c. No menu suspenso Exibir UI, escolha Spark History Server ou Persistent Hive Tez UI para navegar até a interface do aplicativo para seu tipo de trabalho.
   d. Para ver os registros do mecanismo Spark, navegue até a guia Executors na interface do Spark e escolha o link Logs do driver. Para visualizar os registros do mecanismo do Hive, escolha o link Registros apropriado DAG na interface do usuário do Hive Tez.

AWS CLI

Para acessar a interface do usuário do seu aplicativo com o AWS CLI

- Para gerar um URL que você possa usar para acessar a interface do usuário do seu aplicativo para trabalhos em execução e concluídos, chame `GetDashboardForJobRun` API o.

```
aws emr-serverless get-dashboard-for-job-run /
```

```
--application-id <application-id> /
--job-run-id <job-id>
```

O URL que você gera é válido por uma hora.

Monitore as métricas do Spark com o Amazon Managed Service para Prometheus

Com as EMR versões 7.1.0 e posteriores da Amazon, você pode integrar o EMR Serverless ao Amazon Managed Service for Prometheus para coletar métricas do Apache Spark para trabalhos e aplicativos sem servidor. EMR Essa integração está disponível quando você envia um trabalho ou cria um aplicativo usando o AWS console, o EMR Serverless ou API o AWS CLI.

Pré-requisitos

Antes de entregar suas métricas do Spark ao Amazon Managed Service for Prometheus, você deve preencher os seguintes pré-requisitos.

- Crie um espaço de trabalho do Amazon Managed Service para Prometheus. Este Workspace serve como um endpoint de ingestão. Anote o URL exibido para Endpoint - gravação URL remota. Você precisará especificar o URL ao criar seu aplicativo EMR sem servidor.
- Para conceder acesso às suas tarefas ao Amazon Managed Service for Prometheus para fins de monitoramento, adicione a política a seguir à sua função de execução de tarefas.

```
{
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

Configuração

Para usar o AWS console para criar um aplicativo integrado ao Amazon Managed Service for Prometheus

1. Consulte Introdução ao Amazon EMR Serverless para criar um aplicativo.

2. Ao criar um aplicativo, escolha Usar configurações personalizadas e, em seguida, configure seu aplicativo especificando as informações nos campos que você deseja configurar.
3. Em Registros e métricas do aplicativo, escolha Entregar métricas do mecanismo para o Amazon Managed Service for Prometheus e, em seguida, especifique sua gravação remota. URL
4. Especifique qualquer outra configuração desejada e, em seguida, escolha Criar e iniciar aplicativo.

Use o AWS CLI ou sem EMR servidor API

Você também pode usar o AWS CLI ou EMR Serverless API para integrar seu EMR aplicativo Serverless ao Amazon Managed Service for Prometheus quando você estiver executando o ou os comandos. `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"
        }
    }
}'
```

Incluir `prometheusMonitoringConfiguration` em seu comando indica que o EMR Serverless deve executar o trabalho do Spark com um agente que coleta as métricas do Spark e as grava em seu endpoint `remoteWriteUrl` para o Amazon Managed Service for Prometheus. Em seguida, você pode usar as métricas do Spark no Amazon Managed Service for Prometheus para visualização, alertas e análises.

Propriedades avançadas de configuração

EMRO Serverless usa um componente do Spark chamado `PrometheusServlet` para coletar métricas do Spark e traduzir dados de desempenho em dados compatíveis com o Amazon Managed Service for Prometheus. Por padrão, o EMR Serverless define valores padrão no Spark e analisa as métricas do driver e do executor quando você envia um trabalho usando. `PrometheusMonitoringConfiguration`

A tabela a seguir descreve todas as propriedades que você pode configurar ao enviar um trabalho do Spark que envia métricas para o Amazon Managed Service for Prometheus.

Propriedade Spark	Valor padrão	Descrição
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	A classe que o Spark usa para enviar métricas para o Amazon Managed Service for Prometheus. Para substituir o comportamento padrão, especifique sua própria classe personalizada.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source. JvmSource	A classe que o Spark usa para coletar e enviar métricas cruciais da máquina virtual Java subjacente. Para parar de coletar JVM métricas,

Propriedade Spark	Valor padrão	Descrição
		desative essa propriedade definindo-a como uma string vazia, como"". Para substitui r o comportamento padrão, especifique sua própria classe personalizada.
`spark.metrics.conf .driver.sink.prome theusServlet.path`	/métricas/prometheus	O distinto URL que o Amazon Managed Service for Prometheus usa para coletar métricas do motorista . Para substituir o comportam ento padrão, especifique seu próprio caminho. Para parar de coletar métricas do driver, desative essa propriedade definindo-a como uma string vazia, como"".
`spark.metrics.conf .executor.sink.pro metheusServlet.path`	/metrics/executor/prometheus	O distinto URL que o Amazon Managed Service for Prometheus usa para coletar métricas do executor. Para substituir o comportam ento padrão, especifique seu próprio caminho. Para parar de coletar métricas do executor, desative essa propriedade definindo-a como uma string vazia, como"".

Para obter mais informações sobre as métricas do Spark, consulte as métricas do Apache Spark.

Considerações e limitações

Ao usar o Amazon Managed Service for Prometheus para coletar métricas EMR do Serverless, considere as seguintes considerações e limitações.

- O suporte para o uso do Amazon Managed Service para Prometheus EMR com Serverless está disponível somente no Regiões da AWS onde o Amazon Managed Service para Prometheus está disponível ao público em geral.
- Executar o agente para coletar métricas do Spark no Amazon Managed Service para Prometheus exige mais recursos dos trabalhadores. Se você escolher um tamanho de trabalhador menor, como um CPU trabalhador v, o tempo de execução do trabalho poderá aumentar.
- O suporte para o uso do Amazon Managed Service for Prometheus EMR com Serverless está disponível somente para as versões 7.1.0 e superiores da AmazonEMR.

EMRMétricas de uso sem servidor

Você pode usar as métricas CloudWatch de uso da Amazon para dar visibilidade aos recursos que sua conta usa. Use essas métricas para visualizar seu uso do serviço em CloudWatch gráficos e painéis.

EMRAs métricas de uso sem servidor correspondem às Cotas de Serviço. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações, consulte Service Quotas e CloudWatch alarmes da Amazon no Service Quotas User Guide.

Para obter mais informações sobre cotas de serviço EMR sem servidor, consulte. Endpoints e cotas para EMR Serverless

Métricas de uso da cota de serviço para Serverless EMR

EMRO Serverless publica as seguintes métricas de uso da cota de serviço no namespace. `AWS/Usage`

Métrica	Descrição
`ResourceCount`	O número total do recurso especificado que está sendo executado na sua conta. O recurso

Métrica	Descrição
	é definido pelas dimensões associadas à métrica.

Dimensões das métricas de EMR uso da cota de serviço sem servidor

Você pode usar as dimensões a seguir para refinar as métricas de uso publicadas pelo EMR Serverless.

Dimensão	Value	Descrição
`Service`	EMRSem servidor	O nome do AWS service (Serviço da AWS) que contém o recurso.
`Type`	Recurso	O tipo de entidade que o EMR Serverless está relatando.
`Resource`	v CPU	O tipo de recurso que o EMR Serverless está rastreando.
`Class`	Nenhum	A classe de recurso que o EMR Serverless está rastreando.

Automatização sem servidor com EMR Amazon EventBridge

Você pode usar: Amazon EventBridge para automatizar seu Serviços da AWS e responda automaticamente aos eventos do sistema, como problemas de disponibilidade de aplicativos ou alterações de recursos. EventBridge fornece um fluxo quase em tempo real de eventos do sistema que descrevem as mudanças em seu AWS recursos. Você pode escrever regras simples para indicar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. Com EventBridge, você pode automaticamente:

- Invocar um AWS Lambda função
- Transmita um evento para o Amazon Kinesis Data Streams

- Ativar um AWS Step Functions máquina de estado
- Notificar um SNS tópico da Amazon ou uma SQS fila da Amazon

Por exemplo, ao usar EventBridge com EMR Serverless, você pode ativar um AWS Lambda funcione quando um ETL trabalho for bem-sucedido ou notifique um SNS tópico da Amazon quando um ETL trabalho falhar.

EMRO Serverless emite três tipos de eventos:

- Eventos de mudança de estado do aplicativo — Eventos que emitem todas as alterações de estado de um aplicativo. Para obter mais informações sobre os estados do aplicativo, consulteEstados do aplicativo.
- Eventos de mudança de estado de execução de um trabalho — Eventos que emitem cada mudança de estado da execução de um trabalho. Para ter mais informações sobre, consulte Estados de execução de trabalho.
- Eventos de repetição de execução de um trabalho — Eventos que emitem cada nova tentativa de um trabalho executado a partir das versões 7.1.0 e posteriores do Amazon EMR Serverless.

Exemplos de EMR eventos sem servidor EventBridge

Os eventos relatados pelo EMR Serverless têm um valor `aws.emr-serverless` atribuído `asource`, como nos exemplos a seguir.

Evento de alteração do estado do aplicativo

O exemplo de evento a seguir mostra um aplicativo no `CREATING` estado.

```
{
    "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
        }
    }
}
```

Evento de mudança de estado de execução de Job

O exemplo de evento a seguir mostra uma execução de trabalho que se move de um `SCHEDULED` estado para `RUNNING` outro.

```
{
    "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"
```

```
    }
}
```

Evento de repetição de execução de Job

Veja a seguir um exemplo de um evento de repetição de execução de um trabalho.

```
{
    "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"
    }
}
```

Marcando atributos

Você pode atribuir seus próprios metadados a cada recurso usando tags para ajudá-lo a gerenciar seus recursos sem EMR servidor. Esta seção fornece uma visão geral das funções das tags e mostra como criar tags.

Tópicos

O que é uma tag?

Uma tag é um rótulo que você atribui a um AWS recurso. Cada tag consiste em uma chave e um valor, ambos definidos por você. As tags permitem que você categorize seu AWS recursos por atributos como propósito, proprietário e ambiente. Caso possua muitos recursos do mesmo tipo, você pode identificar rapidamente um recurso específico com base nas tags atribuídas a ele. Por exemplo, você pode definir um conjunto de tags para seus aplicativos Amazon EMR Serverless para ajudá-lo a rastrear o proprietário e o nível de pilha de cada aplicativo. Recomendamos planejar um conjunto consistente de chaves de tags para cada tipo de recurso.

Tags não são automaticamente atribuídas aos recursos. Depois de adicionar uma tag a um recurso, você pode modificar o valor de uma tag ou remover a tag do recurso a qualquer momento. As tags não têm nenhum significado semântico para o Amazon EMR Serverless e são interpretadas estritamente como cadeias de caracteres. Se você adicionar uma etiqueta que tenha a mesma chave de uma etiqueta existente nesse recurso, o novo valor substituirá o antigo.

Se você usaIAM, você pode controlar quais usuários em seu AWS a conta tem permissão para gerenciar tags. Para obter exemplos de políticas de controle de acesso por etiquetas, consulte Políticas para controle de acesso baseado em etiquetas.

Marcando seus Recursos

Você pode marcar aplicativos e execuções de tarefas novos ou existentes. Se você estiver usando o Amazon EMR ServerlessAPI, o AWS CLI, ou um AWS SDK, você pode aplicar tags a novos recursos

usando o `tags` parâmetro na API ação relevante. Você pode aplicar tags aos recursos existentes usando a `TagResource` API ação.

Você pode usar algumas ações de criação de recursos para especificar etiquetas para um recurso quando ele for criado. Nesse caso, se as etiquetas não puderem ser aplicadas enquanto o recurso estiver sendo criado, ele não será criado. Esse mecanismo garante que os recursos que você pretende etiquetar na criação sejam criados com as etiquetas especificadas ou não sejam criados. Se você marcar recursos no momento da criação, não precisará executar scripts de marcação personalizados depois de criar um recurso.

A tabela a seguir descreve os recursos do Amazon EMR Serverless que podem ser marcados.

Recurso	Compatível com tags	Compatível com a propagação de tags	Suporta marcação na criação (Amazon EMR API Serverless, AWS CLI e AWS SDK)	APIpara criação (tags podem ser adicionadas durante a criação)
Aplicação	Sim	Não. As tags associadas a um aplicativo não se propagam para execuções de tarefas enviadas a esse aplicativo.	Sim	`CreateApplication`
Execução de trabalho	Sim	Não	Sim	`StartJobRun`

Limitações de marcação

As seguintes limitações básicas se aplicam às tags:

- Cada recurso pode ter no máximo 50 tags criadas pelo usuário.
- Em todos os recursos, cada chave de tag deve ser exclusiva e possuir apenas um valor.

- O tamanho máximo da chave é 128 caracteres Unicode em UTF -8.
- O tamanho máximo do valor é 256 caracteres Unicode em UTF -8.
- Os caracteres permitidos são letras, números, espaços representáveis em UTF -8 e os seguintes caracteres: _.:/= + - @.
- Uma chave de etiqueta não pode ser uma string vazia. Um valor de tag pode ser uma string vazia, mas não nula.
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- Não use `AWS:` nenhuma combinação de maiúsculas ou minúsculas, como prefixo, para chaves ou valores. Eles são reservados apenas para AWS uso.

Trabalhando com tags usando o AWS CLI e o Amazon EMR Serverless API

Use o seguinte AWS CLI comandos ou API operações do Amazon EMR Serverless para adicionar, atualizar, listar e excluir as tags dos seus recursos.

Recurso	Compatível com tags	Compatível com a propagação de tags
Adicione ou substitua uma ou mais tags	`tag-resource`	`TagResource`
Listar as tags para um recurso	`list-tags-for-resource`	`ListTagsForResource`
Exclua uma ou mais tags	`untag-resource`	`UntagResource`

Os exemplos a seguir mostram como marcar ou desmarcar recursos usando o AWS CLI.

Marcar um aplicativo existente

O comando a seguir marca um aplicativo existente.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

Desmarcar um aplicativo existente

O comando a seguir exclui uma tag de um aplicativo existente.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Listar tags para um recurso

O comando a seguir lista as tags associadas a um recurso existente.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

Tutoriais para Serverless EMR

Esta seção descreve casos de uso comuns quando você trabalha com aplicativos EMR sem servidor.

Tópicos

Usando o Java 17 com o Amazon EMR Serverless

Com as EMR versões 6.11.0 e posteriores da Amazon, você pode configurar tarefas do EMR Serverless Spark para usar o tempo de execução do Java 17 para a Java Virtual Machine (). JVM Use um dos métodos a seguir para configurar o Spark com o Java 17.

`JAVA_HOME`

Para substituir a JVM configuração do EMR Serverless 6.11.0 e superior, você pode fornecer a configuração às classificações do Serverless e do `JAVA_HOME` ambiente. `spark.emr-serverless.driverEnv` `spark.executorEnv`

x86_64

Defina as propriedades necessárias para especificar o Java 17 como a `JAVA_HOME` configuração para o driver e os executores do 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

Defina as propriedades necessárias para especificar o Java 17 como a `JAVA_HOME` configuração para o driver e os executores do 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`

Como alternativa, você pode especificar o Java 17 na `spark-defaults` classificação para substituir a JVM configuração do EMR Serverless 6.11.0 e superior.

x86_64

Especifique Java 17 na `spark-defaults` classificação:

```
{
"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

Especifique Java 17 na `spark-defaults` classificação:

```
{
```

```
"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/"
            }
         }
      ]
}
```

Usando o Apache Hudi com o Serverless EMR

Para usar o Apache Hudi com EMR aplicativos sem servidor

1. Defina as propriedades necessárias do Spark na execução da tarefa correspondente do Spark.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Para sincronizar uma tabela Hudi com o catálogo configurado, designe o AWS Glue o Data Catalog como sua metastore ou configure uma metastore externa. EMRO Serverless é compatível com o `hms` modo de sincronização de tabelas do Hive para cargas de trabalho Hudi. EMRO Serverless ativa essa propriedade como padrão. Para saber mais sobre como configurar sua metastore, consulte. Configuração do Metastore

   > **Important**
   >
   > EMRO Serverless não oferece suporte `HIVEQL` ou `JDBC` não oferece opções de modo de sincronização para tabelas do Hive para lidar com cargas de trabalho Hudi. Para saber mais, consulte Modos de sincronização.

   Quando você usa o AWS Glue Data Catalog como sua metastore, você pode especificar as seguintes propriedades de configuração para sua tarefa 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
```

Para saber mais sobre os lançamentos do Apache Hudi da AmazonEMR, consulte o histórico de lançamentos do Hudi.

Usando o Apache Iceberg com o Serverless EMR

Para usar o Apache Iceberg com EMR aplicativos sem servidor

1. Defina as propriedades necessárias do Spark na execução da tarefa correspondente do Spark.

   ```
   spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
   ```

2. Designe ou o AWS Glue o Data Catalog como sua metastore ou configure uma metastore externa. Para saber mais sobre como configurar sua metastore, consulte. Configuração do Metastore

   Configure as propriedades do metastore que você deseja usar para o Iceberg. Por exemplo, se você quiser usar o AWS Glue Data Catalog, defina as seguintes propriedades na configuração do aplicativo.

   ```
   spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-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
   ```

   Quando você usa o AWS Com o Glue Data Catalog como seu metastore, você pode especificar as seguintes propriedades de configuração para seu trabalho no 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://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
   ```

```
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Para saber mais sobre os lançamentos do Apache Iceberg da AmazonEMR, consulte o histórico de lançamentos do Iceberg.

Usando bibliotecas Python com Serverless EMR

Ao executar PySpark trabalhos em aplicativos Amazon EMR Serverless, você pode empacotar várias bibliotecas Python como dependências. Para fazer isso, você pode usar recursos nativos do Python, criar um ambiente virtual ou configurar diretamente seus PySpark trabalhos para usar bibliotecas do Python. Esta página aborda cada abordagem.

Usando recursos nativos do Python

Ao definir a configuração a seguir, você pode usá-la PySpark para carregar arquivos Python (`.py`), pacotes Python compactados () e arquivos Egg `.zip` () para os executores do Spark`.egg`.

```
--conf spark.submit.pyFiles=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/<.py|.egg|.zip file>
```

Para obter mais detalhes sobre como usar ambientes virtuais Python para PySpark trabalhos, consulte Usando recursos PySpark nativos.

Construindo um ambiente virtual Python

Para empacotar várias bibliotecas Python para um PySpark trabalho, você pode criar ambientes virtuais Python isolados.

1. Para criar o ambiente virtual Python, use os comandos a seguir. O exemplo mostrado instala os pacotes `scipy` e `matplotlib` em um pacote de ambiente virtual e copia o arquivo para um local do Amazon S3.

   > **Important**
   >
   > Você deve executar os seguintes comandos em um ambiente Amazon Linux 2 similar com a mesma versão do Python que você usa no EMR Serverless, ou seja, Python 3.7.10 para a Amazon versão 6.6.0. EMR Você pode encontrar um exemplo de Dockerfile no repositório EMRServerless 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://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Envie o trabalho do Spark com suas propriedades definidas para usar o ambiente virtual Python.

```
--conf spark.archives=s3://DOC-EXAMPLE-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
```

Observe que, se você não substituir o binário original do Python, a segunda configuração na sequência de configurações anterior será. `--conf spark.executorEnv.PYSPARK_PYTHON=python`

Para saber mais sobre como usar ambientes virtuais Python para PySpark trabalhos, consulte Usando o Virtualenv. Para ver mais exemplos de como enviar trabalhos do Spark, consulteTrabalhos do Spark.

Configurando PySpark trabalhos para usar bibliotecas Python

Com as EMR versões 6.12.0 e posteriores da Amazon, você pode configurar diretamente PySpark trabalhos EMR sem servidor para usar bibliotecas Python populares de ciência de dados, como pandas, NumPye sem nenhuma configuração adicional. PyArrow

Os exemplos a seguir mostram como empacotar cada biblioteca Python para um PySpark trabalho.

NumPy

NumPy é uma biblioteca Python para computação científica que oferece matrizes e operações multidimensionais para matemática, classificação, simulação aleatória e estatísticas básicas. Para usar NumPy, execute o seguinte comando:

```
import numpy
```

pandas

pandas é uma biblioteca Python construída sobre o. NumPy A biblioteca pandas fornece aos cientistas de dados estruturas de DataFramedados e ferramentas de análise de dados. Para usar pandas, execute o seguinte comando:

```
import pandas
```

PyArrow

PyArrow é uma biblioteca Python que gerencia dados colunares na memória para melhorar o desempenho no trabalho. PyArrow é baseado na especificação de desenvolvimento multilíngue Apache Arrow, que é uma forma padrão de representar e trocar dados em um formato colunar. Para usar PyArrow, execute o seguinte comando:

```
import pyarrow
```

Usando diferentes versões do Python com Serverless EMR

Além do caso de uso emUsando bibliotecas Python com Serverless EMR, você também pode usar ambientes virtuais do Python para trabalhar com versões do Python diferentes da versão empacotada na versão da Amazon para seu aplicativo Amazon EMR Serverless. EMR Para fazer isso, você deve criar um ambiente virtual do Python com a versão do Python que você deseja usar.

Para enviar um trabalho a partir de um ambiente virtual Python

1. Crie seu ambiente virtual com os comandos no exemplo a seguir. Este exemplo instala o Python 3.9.9 em um pacote de ambiente virtual e copia o arquivo para um local do Amazon S3.

> **Important**
>
> Se você usa as EMR versões 7.0.0 e posteriores da Amazon, você deve executar seus comandos em um ambiente Amazon Linux 2023 semelhante ao que você usa para seus aplicativos sem EMR servidor.
> Se você usa a versão 6.15.0 ou inferior, você deve executar os seguintes comandos em um ambiente Amazon Linux 2 similar.

```
# 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. Defina suas propriedades para usar o ambiente virtual Python e enviar o trabalho do Spark.

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://DOC-EXAMPLE-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
```

Para saber mais sobre como usar ambientes virtuais Python para PySpark trabalhos, consulte Usando o Virtualenv. Para ver mais exemplos de como enviar trabalhos do Spark, consulteTrabalhos do Spark.

Usando o Delta Lake OSS com o EMR Serverless

EMRVersões 6.9.0 e superiores da Amazon

> **Note**
>
> O Amazon EMR 7.0.0 e versões posteriores usam o Delta Lake 3.0.0, que renomeia o `delta-core.jar` arquivo para. `delta-spark.jar` Se você usa o Amazon EMR 7.0.0 ou superior, certifique-se de especificar `delta-spark.jar` em suas configurações.

O Amazon EMR 6.9.0 e versões posteriores incluem o Delta Lake, então você não precisa mais empacotar o Delta Lake sozinho ou fornecer à `--packages` bandeira suas tarefas sem EMR servidor.

1. Ao enviar trabalhos EMR sem servidor, verifique se você tem as seguintes propriedades de configuração e inclua os seguintes parâmetros no `sparkSubmitParameters` campo.

```
--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. Crie um local `delta_sample.py` para testar a criação e a leitura de uma tabela Delta.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://DOC-EXAMPLE-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. Usar o AWS CLI, faça o upload do `delta_sample.py` arquivo para o seu bucket do Amazon S3. Em seguida, use o `start-job-run` comando para enviar um trabalho para um aplicativo EMR Serverless existente.

```
aws s3 cp delta_sample.py s3://DOC-EXAMPLE-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://DOC-EXAMPLE-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"
            }
        }'
```

Para usar bibliotecas Python com o Delta Lake, você pode adicionar a `delta-core` biblioteca empacotando-a como uma dependência ou usando-a como uma imagem personalizada.

Como alternativa, você pode usar o `SparkContext.addPyFile` para adicionar as bibliotecas Python do `delta-core` JAR arquivo:

```
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])
```

EMRVersões 6.8.0 e inferiores da Amazon

Se você estiver usando o Amazon EMR 6.8.0 ou inferior, siga estas etapas para usar o Delta Lake OSS com seus aplicativos sem EMR servidor.

1. Para criar uma versão de código aberto do Delta Lake que seja compatível com a versão do Spark em seu aplicativo Amazon EMR Serverless, navegue até o Delta GitHub e siga as instruções.
2. Faça o upload das bibliotecas do Delta Lake em um bucket do Amazon S3 em seu Conta da AWS.
3. Ao enviar trabalhos EMR sem servidor na configuração do aplicativo, inclua os JAR arquivos Delta Lake que agora estão em seu bucket.

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
   ```

4. Para garantir que você possa ler e gravar em uma tabela Delta, execute um PySpark teste de amostra.

   ```
   from pyspark import SparkConf, SparkContext
       from pyspark.sql import HiveContext, SparkSession

       import uuid

       conf = SparkConf()
       sc = SparkContext(conf=conf)
       sqlContext = HiveContext(sc)

       url = "s3://DOC-EXAMPLE-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
```

Enviando trabalhos EMR sem servidor do Airflow

O Amazon Provider no Apache Airflow fornece operadores sem EMR servidor. Para obter mais informações sobre operadores, consulte Amazon EMR Serverless Operators na documentação do Apache Airflow.

Você pode usar `EmrServerlessCreateApplicationOperator` para criar um aplicativo Spark ou Hive. Você também pode usar `EmrServerlessStartJobOperator` para iniciar um ou mais trabalhos com seu novo aplicativo.

Para usar o operador com Amazon Managed Workflows para Apache Airflow (MWAA) com Airflow 2.2.2, adicione a seguinte linha ao seu `requirements.txt` arquivo e atualize seu MWAA ambiente para usar o novo arquivo.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Observe que o suporte EMR sem servidor foi adicionado à versão 5.0.0 do provedor Amazon. A versão 6.0.0 é a última versão compatível com o Airflow 2.2.2. Você pode usar versões posteriores com o Airflow 2.4.3 ativado. MWAA

O exemplo abreviado a seguir mostra como criar um aplicativo, executar várias tarefas do Spark e, em seguida, interromper o aplicativo. Um exemplo completo está disponível no repositório EMRServerless Samples GitHub . Para obter detalhes adicionais sobre a `sparkSubmit` configuração, consulteTrabalhos do Spark.

```
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 = "DOC-EXAMPLE-BUCKET"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://DOC-EXAMPLE-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)
```

Usando funções definidas pelo usuário do Hive com Serverless EMR

As funções definidas pelo usuário (UDFs) do Hive permitem criar funções personalizadas para processar registros ou grupos de registros. Neste tutorial, você usará uma amostra UDF com um aplicativo Amazon EMR Serverless preexistente para executar um trabalho que gera um resultado de consulta. Para saber como configurar um aplicativo, consulteComeçando a usar o Amazon EMR Serverless.

Para usar um UDF com EMR Serverless

1. Navegue até o GitHubpara obter uma amostraUDF. Clone o repositório e alterne para a ramificação git que você deseja usar. Atualize o `maven-compiler-plugin` no `pom.xml` arquivo do repositório para ter uma fonte. Atualize também a configuração da versão Java de destino para`1.8`. Execute `mvn package -DskipTests` para criar o JAR arquivo que contém sua amostraUDFs.
2. Depois de criar o JAR arquivo, faça o upload para o bucket do S3 com o comando a seguir.

   ```
   aws s3 cp brickhouse-0.8.2-JS.jar s3://DOC-EXAMPLE-BUCKET/jars/
   ```

3. Crie um arquivo de exemplo para usar uma das UDF funções de amostra. Salve essa consulta como `udf_example.q` e faça o upload para seu bucket do S3.

   ```
   add jar s3://DOC-EXAMPLE-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. Envie o seguinte trabalho do Hive.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/logs/"
        }
    }
}'
```

5. Use o `get-job-run` comando para verificar o estado do seu trabalho. Aguarde até que o estado mude para`SUCCESS`.

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

6. Faça o download dos arquivos de saída com o comando a seguir.

```
aws s3 cp --recursive s3://DOC-EXAMPLE-BUCKET/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

O `stdout.gz` arquivo é semelhante ao seguinte.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

Usando imagens personalizadas com o EMR Serverless

Tópicos

Use uma versão personalizada do Python

Você pode criar uma imagem personalizada para usar uma versão diferente do Python. Para usar o Python versão 3.10 para trabalhos do Spark, por exemplo, execute o seguinte comando:

```
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 will run the image as hadoop
USER hadoop:hadoop
```

Antes de enviar o trabalho do Spark, defina suas propriedades para usar o ambiente virtual Python, da seguinte maneira.

```
--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
```

Use uma versão personalizada do Java

O exemplo a seguir demonstra como criar uma imagem personalizada para usar o Java 11 em seus trabalhos do Spark.

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

USER root

# install JDK 11
RUN sudo amazon-linux-extras install java-openjdk11

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Antes de enviar o trabalho do Spark, defina as propriedades do Spark para usar o Java 11, da seguinte maneira.

```
--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-
```

Crie uma imagem de ciência de dados

O exemplo a seguir mostra como incluir pacotes Python comuns de ciência de dados, como Pandas e. 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 will run the image as hadoop
USER hadoop:hadoop
```

Processamento de dados geoespaciais com o Apache Sedona

O exemplo a seguir mostra como criar uma imagem para incluir o Apache Sedona para processamento geoespacial.

```
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 will run the image as hadoop
USER hadoop:hadoop
```

Usando a integração do Amazon Redshift para o Apache Spark no Amazon Serverless EMR

Com a EMR versão 6.9.0 e posterior da Amazon, cada imagem de lançamento inclui um conector entre o Apache Spark e o Amazon Redshift. Com esse conector, você pode usar o Spark no Amazon EMR Serverless para processar dados armazenados no Amazon Redshift. A integração é baseada no conector de código aberto `spark-redshift`. Para o Amazon EMR Serverless, a integração do Amazon Redshift com o Apache Spark está incluída como uma integração nativa.

Tópicos

- Lançamento de um aplicativo Spark com a integração do Amazon Redshift para o Apache Spark
- Autenticação com a integração do Amazon Redshift para Apache Spark
- Leitura e gravação de e para o Amazon Redshift
- Considerações e limitações ao usar o conector do Spark

Lançamento de um aplicativo Spark com a integração do Amazon Redshift para o Apache Spark

Para usar a integração com o EMR Serverless 6.9.0, você deve passar as dependências necessárias do Spark-Redshift com seu trabalho do Spark. Use `--jars` para incluir bibliotecas relacionadas ao conector Redshift. Para visualizar outros locais de arquivo com suporte pela opção `--jars`, consulte a seção Advanced Dependency Management da documentação do Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

As EMR versões 6.10.0 e superiores da Amazon não exigem a `minimal-json.jar` dependência e instalam automaticamente as outras dependências em cada cluster por padrão. Os exemplos a seguir mostram como iniciar uma aplicação do Spark com a integração do Amazon Redshift para Apache Spark.

Amazon EMR 6.10.0 +

Inicie um trabalho do Spark no Amazon EMR Serverless com a integração do Amazon Redshift para o Apache Spark na versão 6.10.0 e superior do Serverless. EMR

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

Para iniciar um trabalho do Spark no Amazon EMR Serverless com a integração do Amazon Redshift para o Apache Spark na versão 6.9.0 do EMR Serverless, use a opção conforme mostrado no exemplo a seguir. `--jars` Observe que os caminhos listados com a `--jars` opção são os caminhos padrão para os JAR arquivos.

```
--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
```

Autenticação com a integração do Amazon Redshift para Apache Spark

Use AWS Secrets Manager para recuperar credenciais e se conectar ao Amazon Redshift

Você pode se autenticar com segurança no Amazon Redshift armazenando as credenciais no Secrets Manager e fazendo com que a tarefa do Spark chame o para buscá-la: `GetSecretValue` API

```
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
```

Autentique-se no Amazon Redshift com um driver JDBC

Defina nome de usuário e senha dentro do JDBC URL

Você pode autenticar um trabalho do Spark em um cluster do Amazon Redshift especificando o nome e a senha do banco de dados do Amazon Redshift no. JDBC URL

> **Note**
>
> Se você passar as credenciais do banco de dados noURL, qualquer pessoa que tenha acesso ao também URL poderá acessar as credenciais. Este método geralmente não é recomendado porque não é uma opção segura.

Se a segurança não for uma preocupação para seu aplicativo, você pode usar o seguinte formato para definir o nome de usuário e a senha no JDBCURL:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Use a autenticação IAM baseada com a função de execução de tarefas do Amazon EMR Serverless

A partir da versão 6.9.0 do Amazon EMR Serverless, o JDBC driver 2.1 ou superior do Amazon Redshift é incluído no ambiente. Com JDBC o driver 2.1 e superior, você pode especificar JDBC URL e não incluir o nome de usuário e a senha brutos.

Em vez disso, você pode especificar o esquema `jdbc:redshift:iam://`. Isso faz com que o JDBC driver use sua função de execução de tarefas EMR sem servidor para buscar as credenciais automaticamente. Consulte Configurar uma ODBC conexão JDBC ou para usar IAM credenciais no Guia de Gerenciamento do Amazon Redshift para obter mais informações. Um exemplo disso URL é:

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

As seguintes permissões são necessárias para sua função de execução de tarefas quando as condições fornecidas são atendidas:

Permissão	Condições para se tornar obrigatória para o perfil de execução de trabalho
`redshift:GetClusterCredentials`	Obrigatório para que o JDBC motorista busque as credenciais do Amazon Redshift

Permissão	Condições para se tornar obrigatória para o perfil de execução de trabalho
`redshift:DescribeCluster`	Obrigatório se você especificar o cluster do Amazon Redshift e Região da AWS no endpoint JDBC URL em vez de
`redshift-serverless:GetCredentials`	Obrigatório para que o JDBC motorista busque as credenciais do Amazon Redshift Serverless
`redshift-serverless:GetWorkgroup`	Obrigatório se você estiver usando o Amazon Redshift Serverless e estiver especificando o URL em termos de nome do grupo de trabalho e região

Conectando-se ao Amazon Redshift em um ambiente diferente VPC

Ao configurar um cluster provisionado do Amazon Redshift ou um grupo de trabalho Amazon Redshift Serverless em umVPC, você deve configurar a VPC conectividade para que seu aplicativo Amazon EMR Serverless acesse os recursos. Para obter mais informações sobre como configurar a VPC conectividade em um aplicativo EMR sem servidor, consulte. Configurando o acesso VPC

- Se seu cluster provisionado do Amazon Redshift ou grupo de trabalho Amazon Redshift Serverless estiver acessível publicamente, você poderá especificar uma ou mais sub-redes privadas que tenham um gateway conectado ao criar aplicativos sem servidor. NAT EMR
- Se seu cluster provisionado do Amazon Redshift ou grupo de trabalho Amazon Redshift Serverless não estiver acessível publicamente, você deverá criar um VPC endpoint gerenciado do Amazon Redshift para seu cluster do Amazon Redshift, conforme descrito em. Configurando o acesso VPC Como alternativa, você pode criar seu grupo de trabalho Amazon Redshift Serverless conforme descrito em Conectando-se ao Amazon Redshift Serverless no Guia de Gerenciamento do Amazon Redshift. Você deve associar seu cluster ou seu subgrupo às sub-redes privadas que você especifica ao criar seu EMR aplicativo Serverless.

> **Note**
>
> Se você usa autenticação IAM baseada e suas sub-redes privadas para o aplicativo EMR Serverless não têm um NAT gateway conectado, você também deve criar um VPC endpoint

nessas sub-redes para o Amazon Redshift ou o Amazon Redshift Serverless. Dessa forma, o JDBC motorista pode buscar as credenciais.

Leitura e gravação de e para o Amazon Redshift

Os exemplos de código a seguir são usados PySpark para ler e gravar dados de amostra de e para um banco de dados do Amazon Redshift com uma fonte de dados API e com o Spark. SQL

Data source API

Use PySpark para ler e gravar dados de amostra de e para um banco de dados do Amazon Redshift com fonte de dados. API

```
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

Use PySpark para ler e gravar dados de amostra de e para um banco de dados do Amazon Redshift com o Spark. SQL

```
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()
```

Considerações e limitações ao usar o conector do Spark

- Recomendamos que você ative a JDBC conexão do Spark na Amazon com o Amazon EMR Redshift. SSL

- Recomendamos que você gerencie as credenciais para o cluster Amazon Redshift no AWS Secrets Manager como uma prática recomendada. Consulte Usando AWS Secrets Manager para recuperar credenciais para se conectar ao Amazon Redshift, por exemplo.
- Recomendamos que você transmita uma IAM função com o parâmetro `aws_iam_role` para o parâmetro de autenticação do Amazon Redshift.
- No momento, o parâmetro `tempformat` não é compatível com o formato Parquet.
- Os `tempdir` URI pontos para uma localização do Amazon S3. Esse diretório temporário não é limpo automaticamente e, portanto, pode incorrer em custos adicionais.
- Considere as seguintes recomendações para o Amazon Redshift:
  - Recomendamos bloquear o acesso público ao cluster do Amazon Redshift.
  - Recomendamos ativar o registro em log de auditoria do Amazon Redshift.
  - Recomendamos que você ative a Criptografia em repouso do Amazon Redshift.
- Considere as seguintes recomendações para o Amazon S3:
  - Recomendamos que você bloqueie o acesso público aos buckets do Amazon S3.
  - Recomendamos que você use criptografia no lado do servidor do Amazon S3 para criptografar os buckets do Amazon S3 usados.
  - Recomendamos que você use as políticas de ciclo de vida do Amazon S3 para definir as regras de retenção para o bucket do Amazon S3.
  - A Amazon EMR sempre verifica o código importado do código aberto para a imagem. Por motivos de segurança, não oferecemos suporte aos seguintes métodos de autenticação do Spark para o Amazon S3:
    - Configuração AWS chaves de acesso na classificação `hadoop-env` de configuração
    - Codificação AWS chaves de acesso no `tempdir` URI

Para obter mais informações sobre como usar o conector e os parâmetros compatíveis, consulte os seguintes recursos:

- Integração do Amazon Redshift para Apache Spark no Guia de gerenciamento do Amazon Redshift.
- O repositório da comunidade `spark-redshift` no GitHub.

Conectando-se ao DynamoDB com o Amazon Serverless EMR

Neste tutorial, você carrega um subconjunto de dados do Conselho de Nomes Geográficos dos Estados Unidos para um bucket do Amazon S3 e, em seguida, usa o Hive ou o Spark no Amazon Serverless para copiar os dados em uma tabela do EMR Amazon DynamoDB que você pode consultar.

Etapa 1: Carregar dados em um bucket do Amazon S3

Para criar um bucket do Amazon S3, siga as instruções em Criação de um bucket no Guia do usuário do console do Amazon Simple Storage Service. Substitua *DOC-EXAMPLE-BUCKET* as referências a pelo nome do seu bucket recém-criado. Agora, seu aplicativo EMR sem servidor está pronto para executar trabalhos.

1. Faça o download do arquivo de dados de amostra `features.zip` com o comando a seguir.

   ```
   wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
   features.zip
   ```

2. Extraia o `features.txt` arquivo do arquivo e veja as primeiras linhas do arquivo:

   ```
   unzip features.zip
   head features.txt
   ```

   O resultado deve ser semelhante ao seguinte.

   ```
   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
   ```

   Os campos em cada linha aqui indicam um identificador exclusivo, nome, tipo de característica natural, estado, latitude em graus, longitude em graus e altura em pés.

3. Faça upload de seus dados para o Amazon S3

```
aws s3 cp features.txt s3://DOC-EXAMPLE-BUCKET/features/
```

Etapa 2: criar uma tabela do Hive

Use o Apache Spark ou o Hive para criar uma nova tabela do Hive que contém os dados carregados no Amazon S3.

Spark

Para criar uma tabela do Hive com o Spark, execute o comando a seguir.

```
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://DOC-EXAMPLE_BUCKET/features';")
```

Agora você tem uma tabela preenchida do Hive com dados do `features.txt` arquivo. Para verificar se seus dados estão na tabela, execute uma SQL consulta do Spark conforme mostrado no exemplo a seguir.

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Para criar uma tabela do Hive com o Hive, execute o comando a seguir.

```
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://DOC-EXAMPLE-BUCKET/features';
```

Agora você tem uma tabela do Hive que contém dados do `features.txt` arquivo. Para verificar se seus dados estão na tabela, execute uma consulta HiveQL, conforme mostrado no exemplo a seguir.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

Etapa 3: copiar dados para o DynamoDB

Use o Spark ou o Hive para copiar dados para uma nova tabela do DynamoDB.

Spark

Para copiar dados da tabela do Hive que você criou na etapa anterior para o DynamoDB, siga as etapas 1 a 3 em Copiar dados para o DynamoDB. Isso cria uma nova tabela do DynamoDB chamada. `Features` Em seguida, você pode ler os dados diretamente do arquivo de texto e copiá-los para a tabela do DynamoDB, conforme mostra o exemplo a seguir.

```
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://DOC-EXAMPLE-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

Para copiar dados da tabela do Hive que você criou na etapa anterior para o DynamoDB, siga as instruções em Copiar dados para o DynamoDB.

Etapa 4: consultar dados do DynamoDB

Use o Spark ou o Hive para consultar sua tabela do DynamoDB.

Spark

Para consultar dados da tabela do DynamoDB que você criou na etapa anterior, você pode usar o Spark ou o SQL Spark. MapReduce API

Example — Consulte sua tabela do DynamoDB com o Spark SQL

A SQL consulta do Spark a seguir retorna uma lista de todos os tipos de recursos em ordem alfabética.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

A SQL consulta do Spark a seguir retorna uma lista de todos os lagos que começam com a letra 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;")
```

A SQL consulta do Spark a seguir retorna uma lista de todos os estados com pelo menos três características com mais de uma milha.

```
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 — Consulte sua tabela do DynamoDB com o Spark MapReduce API

A MapReduce consulta a seguir retorna uma lista de todos os tipos de recursos em ordem alfabética.

```
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")
```

A MapReduce consulta a seguir retorna uma lista de todos os lagos que começam com a letra 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")
```

A MapReduce consulta a seguir retorna uma lista de todos os estados com pelo menos três características com mais de uma milha.

```
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

Para consultar dados da tabela do DynamoDB que você criou na etapa anterior, siga as instruções em Consultar os dados na tabela do DynamoDB.

Configurar o acesso entre contas

Para configurar o acesso entre contas para o EMR Serverless, conclua as etapas a seguir. No exemplo, `AccountA` é a conta em que você criou seu aplicativo Amazon EMR Serverless e `AccountB` é a conta em que seu Amazon DynamoDB está localizado.

1. Crie uma tabela do DynamoDB em. `AccountB` Para obter mais informações, consulte Etapa 1: Criar uma tabela.
2. Crie uma `Cross-Account-Role-B` IAM função `AccountB` que possa acessar a tabela do DynamoDB.

   a. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.

   b. Escolha Funções e crie uma nova função chamada`Cross-Account-Role-B`. Para obter mais informações sobre como criar IAM funções, consulte Criação de IAM funções no Guia do usuário.

   c. Crie uma IAM política que conceda permissões para acessar a tabela do DynamoDB entre contas. Em seguida, anexe a IAM política `Cross-Account-Role-B` a.

      Veja a seguir uma política que concede acesso a uma tabela do DynamoDB`CrossAccountTable`.

      ```
      {"Version": "2012-10-17",
          "Statement": [
              {"Effect": "Allow",
                  "Action": "dynamodb:*",
                  "Resource": "arn:aws:dynamodb:region:AccountB:table/
      CrossAccountTable"
              }
          ]
      }
      ```

   d. Edite a relação de confiança para o perfil `Cross-Account-Role-B`.

      Para configurar a relação de confiança para a função, escolha a guia Relações de Confiança no IAM console para a função que você criou na Etapa 2: Cross-Account-Role-B.

Selecione Editar relação de confiança e, em seguida, adicione o seguinte documento de política. Este documento permite que `Job-Execution-Role-A` eu `AccountA` assuma essa `Cross-Account-Role-B` função.

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

e. `Job-Execution-Role-A`Conceda `AccountA` com `- STS Assume role` permissões para assumir`Cross-Account-Role-B`.

No IAM console para Conta da AWS `AccountA`, selecione`Job-Execution-Role-A`. Adicione a instrução de política a seguir ao `Job-Execution-Role-A` para permitir a ação `AssumeRole` no perfil `Cross-Account-Role-B`.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
}
```

f. Defina a `dynamodb.customAWSCredentialsProvider` propriedade com valor como `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` na classificação do site principal. Defina a variável de ambiente `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` com o ARN valor de`Cross-Account-Role-B`.

3. Execute o trabalho do Spark ou do Hive usando. `Job-Execution-Role-A`

Considerações

Considerações ao usar o conector do DynamoDB com o Apache Spark

- O Spark SQL não suporta a criação de uma tabela do Hive com a opção de manipulador de armazenamento. Para obter mais informações, consulte Especificação do formato de armazenamento para tabelas do Hive na documentação do Apache Spark.
- O Spark SQL não suporta a `STORED BY` operação com o manipulador de armazenamento. Se você quiser interagir com uma tabela do DynamoDB por meio de uma tabela externa do Hive, use o Hive para criar a tabela primeiro.
- Para traduzir uma consulta em uma consulta do DynamoDB, o conector do DynamoDB usa o predicate pushdown. O predicate pushdown filtra os dados por uma coluna que é mapeada para a chave de partição de uma tabela do DynamoDB. A pressão de predicados só funciona quando você usa o conector com o SparkSQL, e não com o. MapReduce API

Considerações ao usar o conector do DynamoDB com o Apache Hive

Ajustando o número máximo de mapeadores

- Se você usar a `SELECT` consulta para ler dados de uma tabela externa do Hive mapeada para o DynamoDB, o número de tarefas de mapeamento EMR no Serverless será calculado como a taxa de transferência total de leitura configurada para a tabela do DynamoDB, dividida pela taxa de transferência por tarefa de mapeamento. A taxa de transferência padrão por tarefa de mapeamento é 100.
- O trabalho do Hive pode usar o número de tarefas de mapeamento além do número máximo de contêineres configurados por aplicativo EMR sem servidor, dependendo da taxa de transferência de leitura configurada para o DynamoDB. Além disso, uma consulta de longa duração do Hive pode consumir toda a capacidade de leitura provisionada da tabela do DynamoDB. Isso afeta negativamente outros usuários.
- Você pode usar a `dynamodb.max.map.tasks` propriedade para definir um limite superior para tarefas de mapeamento. Você também pode usar essa propriedade para ajustar a quantidade de dados lidos por cada tarefa de mapa com base no tamanho do contêiner da tarefa.
- Você pode definir a `dynamodb.max.map.tasks` propriedade no nível de consulta do Hive ou na `hive-site` classificação do start-job-run comando. Esse valor deve ser igual ou maior que 1. Quando o Hive processa sua consulta, o trabalho resultante do Hive não usa mais do que os valores de `dynamodb.max.map.tasks` quando lê a tabela do DynamoDB.

Ajustando a taxa de transferência de gravação por tarefa

- A taxa de transferência de gravação por tarefa no EMR Serverless é calculada como a taxa de transferência total de gravação configurada para uma tabela do DynamoDB, dividida pelo valor da propriedade. `mapreduce.job.maps` Para o Hive, o valor padrão dessa propriedade é 2. Assim, as duas primeiras tarefas no estágio final do trabalho do Hive podem consumir toda a taxa de transferência de gravação. Isso leva à limitação das gravações de outras tarefas no mesmo trabalho ou em outros trabalhos.
- Para evitar a limitação de gravação, você pode definir o valor da `mapreduce.job.maps` propriedade com base no número de tarefas no estágio final ou na taxa de transferência de gravação que você deseja alocar por tarefa. Defina essa propriedade na `mapred-site` classificação do start-job-run comando em EMR Serverless.

Segurança

Segurança na nuvem em AWS é a maior prioridade. Como um AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que funciona AWS serviços no AWS Nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte do AWS programas de conformidade. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon EMR Serverless, consulte AWS serviços no escopo do programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e regulamentos aplicáveis.

Essa documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon EMR Serverless. Os tópicos deste capítulo mostram como configurar o Amazon EMR Serverless e usar outros AWS serviços para atender aos seus objetivos de segurança e conformidade.

Tópicos

- Resiliência no Amazon Serverless EMR
- Segurança da infraestrutura no Amazon EMR Serverless
- Análise de configuração e vulnerabilidade no Amazon EMR Serverless

Melhores práticas de segurança para Amazon EMR Serverless

O Amazon EMR Serverless fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas melhores práticas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

Aplicação do princípio de privilégio mínimo

EMRO Serverless fornece uma política de acesso granular para aplicativos que usam IAM funções, como funções de execução. Recomendamos que os perfis de execução recebam somente o conjunto de privilégios mínimos obrigatórios para o trabalho, como a cobertura da aplicação e o acesso ao destino do log. Também recomendamos auditar regularmente as permissões dos trabalhos e após qualquer alteração no código da aplicação.

Isolamento de código de uma aplicação não confiável

EMRO Serverless cria um isolamento total da rede entre trabalhos pertencentes a diferentes aplicativos sem EMR servidor. Nos casos em que o isolamento no nível do trabalho é desejado, considere isolar os trabalhos em diferentes EMR aplicativos sem servidor.

Permissões de controle de acesso baseado em funções () RBAC

Os administradores devem controlar rigorosamente as permissões de controle de acesso (RBAC) baseado em função para EMR aplicativos sem servidor.

Proteção de dados

A ferramenta AWS O modelo de responsabilidade compartilhada se aplica à proteção de dados no Amazon EMR Serverless. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todas as AWS Nuvem. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração

e gerenciamento de segurança do AWS serviços que você usa. Para obter mais informações sobre privacidade de dados, consulte Privacidade de dados FAQ. Para obter informações sobre proteção de dados na Europa, consulte o AWS Modelo de responsabilidade compartilhada e postagem no GDPR blog sobre o AWS Blog de segurança.

Para fins de proteção de dados, recomendamos que você proteja AWS credenciais da conta e configure contas individuais com AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use a autenticação multifator (MFA) com cada conta.
- UseSSL/TLSpara se comunicar com AWS recursos. Recomendamos TLS 1.2 ou posterior.
- Configure API e registre as atividades do usuário com AWS CloudTrail.
- Use AWS soluções de criptografia, junto com todos os controles de segurança padrão dentro AWS serviços.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon S3.
- Use as opções de criptografia Amazon EMR Serverless para criptografar dados em repouso e em trânsito.
- Se você precisar de FIPS 140-2 módulos criptográficos validados ao acessar AWS por meio de uma interface de linha de comando ou umaAPI, use um FIPS endpoint. Para obter mais informações sobre os FIPS endpoints disponíveis, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como números de conta dos seus clientes, em campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Amazon EMR Serverless ou outros AWS serviços usando o consoleAPI, AWS CLI, ou AWS SDKs. Todos os dados que você inserir no Amazon EMR Serverless ou em outros serviços podem ser coletados para inclusão nos registros de diagnóstico. Ao fornecer um URL para um servidor externo, não inclua informações de credenciais no URL para validar sua solicitação para esse servidor.

Criptografia em repouso

A criptografia de dados ajuda a impedir que usuários não autorizados leiam dados em um cluster e em sistemas de armazenamento físico de dados associados. Isso inclui dados salvos em mídias

persistentes, conhecidos como dados em repouso, e dados que podem ser interceptados enquanto viajam pela rede, conhecidos como dados em trânsito.

A criptografia de dados requer chaves e certificados. Você pode escolher entre várias opções, incluindo chaves gerenciadas pelo AWS Key Management Service, chaves gerenciadas pelo Amazon S3 e chaves e certificados de fornecedores personalizados que você fornece. Ao usar AWS KMS como seu provedor de chaves, cobranças são cobradas pelo armazenamento e uso de chaves de criptografia. Para ter mais informações, consulte AWS KMS preços.

Antes de especificar as opções de criptografia, decida quais sistemas de gerenciamento de chaves e de certificados você deseja usar. Em seguida, crie as chaves e os certificados para os provedores personalizados especificados como parte das configurações de criptografia.

Criptografia em repouso para EMRFS dados no Amazon S3

Cada aplicativo EMR sem servidor usa uma versão de lançamento específica, que inclui EMRFS (Sistema de EMR arquivos). A criptografia do Amazon S3 funciona com objetos EMR do Sistema de Arquivos (EMRFS) lidos e gravados no Amazon S3. Você pode especificar a criptografia do lado do servidor (SSE) ou do cliente () do Amazon S3 como o modo de criptografia padrão ao CSE ativar a criptografia em repouso. Opcionalmente, você pode especificar diferentes métodos de criptografia para buckets individuais usando Per bucket encryption overrides (Substituições de criptografia por bucket). Independentemente de a criptografia do Amazon S3 estar habilitada, o Transport Layer Security (TLS) criptografa os EMRFS objetos em trânsito entre os nós do EMR cluster e o Amazon S3. Se você usa o Amazon S3 CSE com chaves gerenciadas pelo cliente, sua função de execução usada para executar trabalhos em um aplicativo EMR sem servidor deve ter acesso à chave. Para obter informações detalhadas sobre a criptografia do Amazon S3, consulte Proteção de dados usando criptografia no Guia do desenvolvedor do Amazon Simple Storage Service.

> **Note**
>
> Quando você usa AWS KMS, cobranças são cobradas pelo armazenamento e uso de chaves de criptografia. Para ter mais informações, consulte AWS KMS preços.

Criptografia do lado do servidor do Amazon S3

Quando você configura a criptografia do lado do servidor do Amazon S3, o Amazon S3 criptografa os dados no nível do objeto à medida que os grava no disco e os descriptografa quando são acessados.

Para obter mais informações sobre issoSSE, consulte Proteção de dados usando criptografia do lado do servidor no Guia do desenvolvedor do Amazon Simple Storage Service.

Você pode escolher entre dois sistemas de gerenciamento de chaves diferentes ao especificar SSE no Amazon EMR Serverless:

- SSE-S3 - O Amazon S3 gerencia as chaves para você. Nenhuma configuração adicional é necessária no EMR Serverless.
- SSE- KMS - Você usa um AWS KMS key para configurar com políticas adequadas para EMR Serverless. Nenhuma configuração adicional é necessária no EMR Serverless.

Para usar AWS KMS criptografia para dados que você grava no Amazon S3, você tem duas opções ao usar o. `StartJobRun` API Você pode habilitar a criptografia para tudo o que você grava no Amazon S3, ou você pode habilitar a criptografia para dados que você grava em um bucket específico. Para obter mais informações sobre o `StartJobRun`API, consulte a Referência EMRsem servidor API.

Para ativar AWS KMS criptografia para todos os dados que você grava no Amazon S3, use os seguintes comandos ao chamar o. `StartJobRun` API

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Para ativar AWS KMS criptografia para dados que você grava em um bucket específico, use os comandos a seguir ao chamar `StartJobRun` API o.

```
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-BUCKET>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-
BUCKET>.serverSideEncryption.kms.keyId=<kms-id>
```

SSEcom chaves fornecidas pelo cliente (SSE-C) não está disponível para uso com Serverless. EMR

Criptografia do lado do cliente do Amazon S3

Com a criptografia do lado do cliente do Amazon S3, a criptografia e a descriptografia do Amazon S3 ocorrem no cliente disponível em cada versão da Amazon. EMRFS EMR Os objetos são criptografados antes de serem carregados no Amazon S3 e descriptografados após serem baixados. O provedor especificado por você fornece a chave de criptografia que o cliente usa. O cliente pode

usar as chaves fornecidas pelo AWS KMS (CSE-KMS) ou uma classe Java personalizada que fornece a chave raiz do lado do cliente (CSE-C). As especificações da criptografia são ligeiramente diferentes entre CSE - KMS e CSE -C, dependendo do provedor especificado e dos metadados do objeto que está sendo descriptografado ou criptografado. Se você usa o Amazon S3 CSE com chaves gerenciadas pelo cliente, sua função de execução usada para executar trabalhos em um aplicativo EMR sem servidor deve ter acesso à chave. KMSTaxas adicionais podem ser aplicadas. Para obter mais informações sobre essas diferenças, consulte Proteção de dados usando criptografia do lado do cliente no Guia do desenvolvedor do Amazon Simple Storage Service.

Criptografia de disco local

Os dados armazenados em armazenamento temporário são criptografados com chaves de propriedade do serviço usando o algoritmo criptográfico AES -256 padrão do setor.

Gerenciamento de chaves

Você pode configurar KMS para girar automaticamente suas KMS chaves. Isso alterna suas chaves uma vez por ano, enquanto salva as chaves antigas indefinidamente para que seus dados ainda possam ser descriptografados. Para obter informações adicionais, consulte Chaves mestras rotativas do cliente.

Criptografia em trânsito

Os seguintes recursos de criptografia específicos do aplicativo estão disponíveis com o Amazon Serverless: EMR

- Spark
  - Por padrão, a comunicação entre drivers e executores do Spark é autenticada e interna. RPCa comunicação entre drivers e executores é criptografada.
- Hive
  - Comunicação entre o AWS O metastore Glue e os aplicativos EMR sem servidor acontecem via. TLS

Você deve permitir somente conexões criptografadas sobre HTTPS (TLS) usando a SecureTransport condição aws: nas políticas de bucket IAM do Amazon S3.

Identity and Access Management (IAM) no Amazon EMR Serverless

AWS Identity and Access Management (IAM) é um AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso ao AWS recursos. IAMos administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar os recursos do Amazon EMR Serverless. IAMé um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

Como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Amazon EMR Serverless.

Usuário do serviço — Se você usa o serviço Amazon EMR Serverless para fazer seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do Amazon EMR Serverless para fazer seu trabalho, você pode precisar de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um recurso no Amazon EMR Serverless, consulte. Solução de problemas de identidade e acesso ao Amazon EMR Serverless

Administrador de serviços — Se você é responsável pelos recursos do Amazon EMR Serverless em sua empresa, provavelmente tem acesso total ao Amazon EMR Serverless. É seu trabalho determinar quais recursos e recursos do Amazon EMR Serverless seus usuários do serviço devem acessar. Em seguida, você deve enviar solicitações ao IAM administrador para alterar as permissões dos usuários do serviço. Revise as informações nesta página para entender os conceitos básicos doIAM. Para saber mais sobre como sua empresa pode usar o IAM Amazon EMR Serverless, consulte. Identity and Access Management (IAM) no Amazon EMR Serverless

IAMadministrador — Se você for IAM administrador, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao Amazon EMR Serverless. Para ver exemplos de políticas baseadas em identidade do Amazon EMR Serverless que você pode usar, consulte. IAM Exemplos de políticas baseadas em identidade para Serverless EMR

Autenticando com identidades

Autenticação é como você faz login em AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado em AWS) como o Usuário raiz da conta da AWS, como IAM usuário ou assumindo uma IAM função.

Você pode fazer login em AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Os usuários (do IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você entra como uma identidade federada, seu administrador configurou previamente a federação de identidades usando IAM funções. Quando você acessa AWS ao usar a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou o AWS portal de acesso. Para obter mais informações sobre como fazer login no AWS, veja Como fazer login no seu Conta da AWS no Início de Sessão da AWS Guia do usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para você mesmo assinar solicitações, consulte Assinatura AWS APIsolicitações no Guia do IAM usuário.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação

multifator no AWS IAM Identity Center Guia do usuário e uso da autenticação multifatorial (MFA) em AWS no IAM Guia do usuário.

Conta da AWS usuário raiz

Quando você cria um Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS e recursos na conta. Essa identidade é chamada de Conta da AWS usuário root e é acessado fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para ver a lista completa de tarefas que exigem que você faça login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do IAM usuário.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, um provedor de identidade da web, o AWS Directory Service, o diretório do Identity Center ou qualquer usuário que acesse Serviços da AWS usando credenciais fornecidas por meio de uma fonte de identidade. Quando as identidades federadas acessam Contas da AWS, eles assumem funções, e as funções fornecem credenciais temporárias.

Para gerenciamento de acesso centralizado, recomendamos que você use AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todas as suas Contas da AWS e aplicativos. Para obter informações sobre o IAM Identity Center, consulte O que é o IAM Identity Center? no AWS IAM Identity Center Guia do usuário.

Grupos e usuários do IAM

Um IAMusuário é uma identidade dentro do seu Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos confiar em credenciais temporárias em vez de criar IAM usuários que tenham credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com IAM os usuários, recomendamos que você alterne as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exigem credenciais de longo prazo no Guia do IAMusuário.

Um IAMgrupo é uma identidade que especifica uma coleção de IAM usuários. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar IAM recursos.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um IAM usuário (em vez de uma função) no Guia do IAM usuário.

IAMfunções

Um IAMpapel é uma identidade dentro de você Conta da AWS que tem permissões específicas. É semelhante a um IAM usuário, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma IAM função no AWS Management Console trocando de papéis. Você pode assumir uma função chamando um AWS CLI ou AWS APIoperação ou usando um personalizadoURL. Para obter mais informações sobre métodos de uso de funções, consulte Usando IAM funções no Guia IAM do usuário.

IAMfunções com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter informações sobre funções para federação, consulte Criação de uma função para um provedor de identidade terceirizado no Guia IAM do usuário. Se você usa o IAM Identity Center, configura um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a uma função em. IAM Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no AWS IAM Identity Center Guia do usuário.
- Permissões temporárias IAM de IAM usuário — Um usuário ou função pode assumir uma IAM função para assumir temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas — Você pode usar uma IAM função para permitir que alguém (um diretor confiável) em uma conta diferente acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a

diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

- Acesso entre serviços — Alguns Serviços da AWS use recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um IAM usuário ou função para realizar ações no AWS, você é considerado um diretor. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinado com a solicitação AWS service (Serviço da AWS) para fazer solicitações para serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou recursos a serem concluídos. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.
  - Função de serviço — Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.
  - Função vinculada a serviços — Uma função vinculada a serviços é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode assumir a função de realizar uma ação em seu nome. As funções vinculadas ao serviço aparecem em seu Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.
- Aplicativos em execução na Amazon EC2 — Você pode usar uma IAM função para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo AWS CLI ou AWS APIsolicitações. Isso é preferível ao armazenamento de chaves de acesso na EC2 instância. Para atribuir um AWS Ao atribuir a uma EC2 instância e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância que é anexado à instância. Um perfil de instância contém a função e permite que programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Como usar uma IAM função para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia IAM do usuário.

Para saber se usar IAM funções ou IAM usuários, consulte Quando criar uma IAM função (em vez de um usuário) no Guia do IAM usuário.

Gerenciando acesso usando políticas

Você controla o acesso em AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto em AWS que, quando associados a uma identidade ou recurso, definem suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada em AWS como JSON documentos. Para obter mais informações sobre a estrutura e o conteúdo dos documentos de JSON política, consulte Visão geral das JSON políticas no Guia IAM do usuário.

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

IAMas políticas definem permissões para uma ação, independentemente do método usado para realizar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função do AWS Management Console, o AWS CLI, ou o AWS API.

Políticas baseadas em identidade

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas gerenciadas incluem AWS políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma

política gerenciada ou uma política em linha, consulte Escolha entre políticas gerenciadas e políticas em linha no Guia do IAMusuário.

Políticas baseadas no recurso

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou Serviços da AWS.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar AWS políticas gerenciadas a partir IAM de uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Amazon S3, AWS WAF, e a Amazon VPC são exemplos de serviços que oferecem suporteACLs. Para saber maisACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões — Um limite de permissões é um recurso avançado no qual você define as permissões máximas que uma política baseada em identidade pode conceder a uma IAM entidade (IAMusuário ou função). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações

sobre limites de permissões, consulte Limites de permissões para IAM entidades no Guia IAM do usuário.

- Políticas de controle de serviço (SCPs) — SCPs são JSON políticas que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. Os SCP limites de permissões para entidades em contas de membros, incluindo cada Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations eSCPs, consulte Políticas de controle de serviços no AWS Organizations Guia do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia IAM do usuário.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determina se uma solicitação deve ser permitida quando vários tipos de política estão envolvidos, consulte Lógica de avaliação de políticas no Guia IAM do usuário.

Como o EMR Serverless funciona com IAM

Antes de usar IAM para gerenciar o acesso ao Amazon EMR Serverless, saiba quais IAM recursos estão disponíveis para uso com o Amazon EMR Serverless.

IAMrecursos que você pode usar com o EMR Serverless

IAMrecurso	Suporte ao Amazon EMR Serverless
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não

IAMrecurso	Suporte ao Amazon EMR Serverless
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de políticas	Não
ACLs	Não
ABAC(tags nas políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Não
Funções vinculadas ao serviço	Sim

Para obter uma visão de alto nível de como o EMR Serverless e outros AWS os serviços funcionam com a maioria dos IAM recursos, consulte AWS serviços que funcionam com IAM o Guia IAM do Usuário.

Políticas baseadas em identidade para servidores sem servidor EMR

Compatível com políticas baseadas em identidade: Sim

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

Com políticas IAM baseadas em identidade, você pode especificar ações e recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que você pode usar em uma JSON política, consulte a referência IAM JSON de elementos de política no Guia IAM do usuário.

Exemplos de políticas baseadas em identidade para Serverless EMR

Para ver exemplos de políticas baseadas em identidade do Amazon EMR Serverless, consulte. Exemplos de políticas baseadas em identidade para Serverless EMR

Políticas baseadas em recursos dentro do Serverless EMR

Suporte a políticas baseadas em recursos: não

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou Serviços da AWS.

Para habilitar o acesso entre contas, você pode especificar uma conta ou IAM entidades inteiras em outra conta como principal em uma política baseada em recursos. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso estão em condições diferentes Contas da AWS, um IAM administrador na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Ações políticas para EMR Serverless

Compatível com ações de políticas: Sim

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O `Action` elemento de uma JSON política descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome que as associadas AWS APIoperação. Há algumas exceções, como ações somente de permissão que não

têm uma operação correspondente. API Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de ações EMR sem servidor, consulte Ações, recursos e chaves de condição do Amazon EMR Serverless na Referência de autorização de serviço.

As ações de política no EMR Serverless usam o seguinte prefixo antes da ação.

```
emr-serverless
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Para ver exemplos de políticas baseadas em identidade do Amazon EMR Serverless, consulte. Exemplos de políticas baseadas em identidade para Serverless EMR

Recursos de políticas para EMR Serverless

Compatível com recursos de políticas: Sim

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` JSON de política especifica o objeto ou objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu Amazon Resource Name (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon EMR Serverless e seusARNs, consulte Recursos definidos pelo Amazon EMR Serverless na Referência de autorização de serviço. Para saber quais ações você pode especificar para cada recurso, consulte Ações, recursos e chaves de condição para Amazon EMR Serverless. ARN

Para ver exemplos de políticas baseadas em identidade do Amazon EMR Serverless, consulte. Exemplos de políticas baseadas em identidade para Serverless EMR

Chaves de condição de política para EMR Serverless

Suporta chaves de condição de política específicas de serviço	Não

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários `Condition` elementos em uma instrução ou várias chaves em um único `Condition` elemento, AWS os avalia usando uma `AND` operação lógica. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, você pode conceder permissão a um IAM usuário para acessar um recurso somente se ele estiver marcado com o nome de IAM usuário. Para obter mais informações, consulte elementos de IAM política: variáveis e tags no Guia IAM do usuário.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver tudo AWS chaves de condição globais, consulte AWS chaves de contexto de condição global no Guia IAM do usuário.

Para ver uma lista das chaves de condição do Amazon EMR Serverless e saber quais ações e recursos você pode usar uma chave de condição, consulte Ações, recursos e chaves de condição do Amazon EMR Serverless na Referência de autorização de serviço.

Todas as EC2 ações da Amazon oferecem suporte às chaves de `ec2:Region` condição `aws:RequestedRegion` e de condição. Para obter mais informações, consulte Exemplo: restringir o acesso a uma região específica.

Listas de controle de acesso (ACLs) em EMR Serverless

SuportesACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Controle de acesso baseado em atributos (ABAC) com Serverless EMR

Suportes ABAC (tags nas políticas)	Sim

O controle de acesso baseado em atributos (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a IAM entidades (usuários ou funções) e a muitas AWS recursos. Marcar entidades e recursos é a primeira etapa doABAC. Em seguida, você cria ABAC políticas para permitir operações quando a tag do diretor corresponde à tag do recurso que ele está tentando acessar.

ABACé útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna complicado.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobreABAC, consulte O que éABAC? no Guia do IAM usuário. Para ver um tutorial com etapas de configuraçãoABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do IAMusuário.

Usando credenciais temporárias com Serverless EMR

Compatível com credenciais temporárias: Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS trabalhar com credenciais temporárias, consulte Serviços da AWS que funcionam com IAM o Guia IAM do Usuário.

Você está usando credenciais temporárias se fizer login no AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre a troca de funções, consulte Alternando para uma função (console) no Guia IAM do usuário.

Você pode criar manualmente credenciais temporárias usando o AWS CLI ou AWS API. Você pode então usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias emIAM.

Permissões principais entre serviços para Serverless EMR

Suporta sessões de acesso direto (FAS): Sim

Quando você usa um IAM usuário ou uma função para realizar ações no AWS, você é considerado um diretor. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinado com a solicitação AWS service (Serviço da AWS) para fazer solicitações para serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou recursos para concluir. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.

Funções de serviço para EMR Serverless

Oferece suporte a perfis de serviço	Não

Funções vinculadas a serviços para Serverless EMR

Oferece suporte a perfis vinculados ao serviço	Sim

Para obter detalhes sobre a criação ou o gerenciamento de funções vinculadas ao serviço, consulte AWS serviços que funcionam com IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Usando funções vinculadas a serviços para Serverless EMR

Usos do Amazon EMR Serverless AWS Identity and Access Management (IAM) funções vinculadas ao serviço. Uma função vinculada a serviços é um tipo exclusivo de IAM função vinculada diretamente ao EMR Serverless. As funções vinculadas ao serviço são predefinidas pelo EMR Serverless e incluem todas as permissões que o serviço exige para chamar outras AWS serviços em seu nome.

Uma função vinculada ao serviço facilita a configuração do EMR Serverless porque você não precisa adicionar manualmente as permissões necessárias. EMRO Serverless define as permissões de suas funções vinculadas ao serviço e, a menos que seja definido de outra forma, somente o EMR Serverless pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política de permissões não pode ser anexada a nenhuma outra IAM entidade.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos EMR sem servidor porque você não pode remover inadvertidamente a permissão para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a funções vinculadas a serviços, consulte AWS Serviços que funcionam com IAM e procure os serviços que têm Sim na coluna Funções vinculadas ao serviço. Escolha um Sim com um link para visualizar a documentação da função vinculada a esse serviço.

Permissões de função vinculadas a serviços para Serverless EMR

EMRO Serverless usa a função vinculada ao serviço chamada AWSServiceRoleForAmazonEMRServerlesspara permitir que ele chame AWS APIsem seu nome.

A função AWSServiceRoleForAmazonEMRServerless vinculada ao serviço confia nos seguintes serviços para assumir a função:

- `ops.emr-serverless.amazonaws.com`

A política de permissões de função nomeada `AmazonEMRServerlessServiceRolePolicy` permite que o EMR Serverless conclua as seguintes ações nos recursos especificados.

> **Note**
>
> O conteúdo da política gerenciada muda, então a política mostrada aqui pode estar desatualizada. Veja a maior parte up-to-date da política A mazonEMRServerless ServiceRolePolicy no AWS Management Console.

- Ação: `ec2:CreateNetworkInterface`
- Ação: `ec2:DeleteNetworkInterface`
- Ação: `ec2:DescribeNetworkInterfaces`
- Ação: `ec2:DescribeSecurityGroups`
- Ação: `ec2:DescribeSubnets`
- Ação: `ec2:DescribeVpcs`
- Ação: `ec2:DescribeDhcpOptions`
- Ação: `ec2:DescribeRouteTables`
- Ação: `cloudwatch:PutMetricData`

A seguir está a `AmazonEMRServerlessServiceRolePolicy` política completa.

```
{
    "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"
                    ]
                }
            }
        }
    ]
}
```

A política de confiança a seguir está anexada a essa função para permitir que o diretor EMR sem servidor assuma essa função.

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

Você deve configurar permissões para permitir que uma IAM entidade (como um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte Permissões de funções vinculadas ao serviço no Guia do IAMusuário.

Criação de uma função vinculada a serviços para Serverless EMR

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria um novo aplicativo EMR sem servidor no AWS Management Console (usando o EMR Studio), o AWS CLI, ou o AWS API, o EMR Serverless cria a função vinculada ao serviço para você. Você deve configurar permissões para permitir que uma IAM entidade (como um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço.

Para criar a função AWSServiceRoleForAmazonEMRServerless vinculada ao serviço usando IAM

Adicione a seguinte declaração à política de permissões da IAM entidade que precisa criar a função vinculada ao serviço.

```
{
    "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"}}
}
```

Se excluir essa função vinculada ao serviço e precisar criá-la novamente, você poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria um novo aplicativo EMR Serverless, o EMR Serverless cria a função vinculada ao serviço para você novamente.

Você também pode usar o IAM console para criar uma função vinculada ao serviço com o caso de uso sem EMRservidor. No painel, AWS CLI ou o AWS API, crie uma função vinculada ao serviço com o nome do `ops.emr-serverless.amazonaws.com` serviço. Para obter mais informações, consulte Criação de uma função vinculada ao serviço no Guia do IAMusuário. Se você excluir essa função vinculada ao serviço, será possível usar esse mesmo processo para criar a função novamente.

Editando uma função vinculada ao serviço para Serverless EMR

EMRO Serverless não permite que você edite a função AWSServiceRoleForAmazonEMRServerless vinculada ao serviço porque várias entidades podem fazer referência à função. Você não pode editar o AWS IAM-política de propriedade que a função vinculada ao serviço EMR Serverless usa, pois contém todas as permissões necessárias que o Serverless precisa. EMR No entanto, você pode editar a descrição da função usandoIAM.

Para editar a descrição da função AWSServiceRoleForAmazonEMRServerless vinculada ao serviço usando IAM

Adicione a declaração a seguir à política de permissões da IAM entidade que precisa editar a descrição de uma função vinculada ao serviço.

```
{
    "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"}}
}
```

Para obter mais informações, consulte Edição de uma função vinculada ao serviço no Guia do IAMusuário.

Excluindo uma função vinculada ao serviço para Serverless EMR

Se você não precisar mais usar um atributo ou serviço que exija uma função vinculada a um serviço, recomendamos que você exclua essa função. Isso é para que você não tenha uma entidade não utilizada que não seja monitorada ou mantida ativamente. No entanto, você deve excluir todos os aplicativos EMR sem servidor em todas as regiões antes de excluir a função vinculada ao serviço.

> **Note**
>
> Se o serviço EMR Serverless estiver usando a função quando você tentar excluir os recursos associados à função, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir a função AWSServiceRoleForAmazonEMRServerless vinculada ao serviço usando IAM

Adicione a declaração a seguir à política de permissões da IAM entidade que precisa excluir uma função vinculada ao serviço.

```
{
    "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"}}
}
```

Para excluir manualmente a função vinculada ao serviço usando IAM

Use o IAM console, o AWS CLI, ou o AWS APIpara excluir a função AWSServiceRoleForAmazonEMRServerless vinculada ao serviço. Para obter mais informações, consulte Excluindo uma função vinculada ao serviço no Guia do IAM usuário.

Regiões suportadas para funções vinculadas a serviços EMR sem servidor

EMRO Serverless oferece suporte ao uso de funções vinculadas ao serviço em todas as regiões em que o serviço está disponível. Para ter mais informações, consulte AWS Regiões e endpoints.

Funções de tempo de execução de trabalho para Amazon EMR Serverless

Você pode especificar as permissões de IAM função que uma execução de trabalho EMR sem servidor pode assumir ao chamar outros serviços em seu nome. Isso inclui acesso ao Amazon S3 para quaisquer fontes de dados, destinos e outros AWS recursos como clusters do Amazon Redshift e tabelas do DynamoDB. Para saber mais sobre como criar uma função, consulteCrie uma função de tempo de execução do trabalho.

Exemplos de políticas de tempo de execução

Você pode anexar uma política de tempo de execução, como a seguinte, a uma função de tempo de execução do trabalho. A seguinte política de tempo de execução de tarefas permite:

- Acesso de leitura aos buckets do Amazon S3 com amostras. EMR
- Acesso total aos buckets do S3.
- Crie e leia o acesso ao AWS Catálogo de dados Glue.

Para adicionar acesso a outros AWS recursos como o DynamoDB, você precisará incluir permissões para eles na política ao criar a função de tempo de execução.

```
{
    "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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-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": ["*"]
        }
    ]
}
```

Passe privilégios de função

Você pode anexar políticas de IAM permissões à função de um usuário para permitir que ele transmita somente funções aprovadas. Isso permite que os administradores controlem quais usuários podem passar funções específicas de tempo de execução de tarefas para tarefas EMR sem servidor. Para saber mais sobre como definir permissões, consulte Conceder permissões a um usuário para passar uma função para um AWS serviço.

Veja a seguir um exemplo de política que permite passar uma função de tempo de execução do trabalho para o responsável pelo serviço EMR sem servidor.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

Exemplos de políticas de acesso de usuários para EMR Serverless

Você pode configurar políticas refinadas para seus usuários, dependendo das ações que você deseja que cada usuário execute ao interagir com aplicativos sem servidor. EMR As políticas a seguir são exemplos que podem ajudar na configuração das permissões certas para seus usuários. Esta seção se concentra somente nas EMR políticas sem servidor. Para exemplos de políticas de usuário do EMR Studio, consulte Configurar permissões de usuário do EMR Studio. Para obter informações sobre como anexar políticas aos IAM usuários (diretores), consulte Gerenciamento de IAM políticas no Guia do IAM usuário.

Política de usuários avançados

Para conceder todas as ações necessárias para o EMR Serverless, crie e anexe uma `AmazonEMRServerlessFullAccess` política ao IAM usuário, função ou grupo necessário.

Veja a seguir um exemplo de política que permite que usuários avançados criem e modifiquem aplicativos EMR sem servidor, além de realizar outras ações, como enviar e depurar trabalhos. Ele revela todas as ações que o EMR Serverless exige para outros serviços.

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

```
}
```

Quando você ativa a conectividade de rede com o seuVPC, os aplicativos EMR sem servidor criam interfaces de rede EC2 elástica da Amazon (ENIs) para se comunicar com VPC os recursos. A política a seguir garante que qualquer novo EC2 ENIs seja criado somente no contexto de aplicativos EMR sem servidor.

> **Note**
>
> É altamente recomendável definir essa política para garantir que os usuários não possam criar EC2ENIs, exceto no contexto da inicialização de aplicativos EMR sem servidor.

```
{
    "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"
                }
            }
        }
}
```

Se quiser restringir o acesso EMR sem servidor a determinadas sub-redes, você pode marcar cada sub-rede com uma condição de tag. Essa IAM política garante que os aplicativos EMR sem servidor só possam ser criados EC2 ENIs dentro de sub-redes permitidas.

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
```

```
            "ec2:CreateNetworkInterface"
        ],
        "Resource": [
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:security-group/*"
        ],
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/KEY": "VALUE"
            }
        }
}
```

> **Important**
>
> Se você for um administrador ou usuário avançado criando seu primeiro aplicativo, deverá configurar suas políticas de permissão para permitir a criação de uma função vinculada a serviços EMR sem servidor. Para saber mais, consulte Usando funções vinculadas a serviços para Serverless EMR.

A IAM política a seguir permite que você crie uma função vinculada ao serviço EMR sem servidor para sua conta.

```
{
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

Política de engenharia de dados

Veja a seguir um exemplo de política que permite aos usuários permissões somente de leitura em aplicativos EMR sem servidor, bem como a capacidade de enviar e depurar trabalhos. Lembre-se de que, como essa política não nega explicitamente as ações, uma declaração de política diferente ainda pode ser usada para conceder acesso a ações específicas.

```
{
```

```
    "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": "*"
        }
    ]
}
```

Uso de tags para controle de acesso com

Você pode usar condições de tag para um controle de acesso refinado. Por exemplo, você pode restringir os usuários de uma equipe para que eles só possam enviar trabalhos para aplicativos EMR sem servidor marcados com o nome da equipe.

```
{
    "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": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:ResourceTag/Team": "team-name"
                }
            }
        }
    ]
}
```

Políticas para controle de acesso baseado em etiquetas

Você pode usar condições em sua política baseada em identidade para controlar o acesso a aplicativos e execuções de trabalhos com base em tags.

Os exemplos a seguir demonstram diferentes cenários e formas de usar operadores de condição com chaves de condição EMR sem servidor. Essas declarações IAM de política são destinadas apenas para fins de demonstração e não devem ser usadas em ambientes de produção. Há várias maneiras de combinar declarações de políticas para conceder e negar permissões de acordo com seus requisitos. Para obter mais informações sobre IAM políticas de planejamento e teste, consulte o Guia IAM do usuário.

> **Important**
>
> Recusar, explicitamente, permissões para ações de uso de tags é uma consideração importante. Isso evita que os usuários façam a marcação de um recurso e, assim, concedam a si mesmos permissões que você não pretendia conceder. Se as ações de marcação de um recurso não forem negadas, um usuário poderá modificar as etiquetas e driblar a intenção das políticas baseadas em etiquetas. Para obter um exemplo de política que nega ações de marcação, consulte Negação de acesso para adição ou remoção de etiquetas.

Os exemplos abaixo demonstram políticas de permissões baseadas em identidade que são usadas para controlar as ações permitidas com aplicativos sem EMR servidor.

Ações permitidas somente em recursos com valores de etiquetas específicos

No exemplo de política a seguir, o operador de `StringEquals` condição tenta corresponder `dev` ao valor do departamento de tag. Se o departamento de tags não tiver sido adicionado ao aplicativo ou não contiver o valor`dev`, a política não se aplica e as ações não são permitidas por essa política. Se nenhuma outra declaração de política permitir as ações, o usuário só poderá trabalhar com aplicativos que tenham essa tag com esse valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Você também pode especificar vários valores de tag usando um operador de condição. Por exemplo, para permitir ações em aplicativos nos quais a `department` tag contém o valor `dev` ou`test`, você pode substituir o bloco de condições no exemplo anterior pelo seguinte.

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

Marcação obrigatória na criação de um recurso

No exemplo abaixo, a tag precisa ser aplicada ao criar o aplicativo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
```

```
      "Condition": {
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

A declaração de política a seguir permite que um usuário crie um aplicativo somente se o aplicativo tiver uma `department` tag que possa conter qualquer valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "emr-serverless:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

Negação de acesso para adição ou remoção de etiquetas

Essa política impede que um usuário adicione ou remova tags em aplicativos EMR sem servidor com uma `department` tag cujo valor não seja. `dev`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
```

```
        "emr-serverless:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Exemplos de políticas baseadas em identidade para Serverless EMR

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do Amazon EMR Serverless. Eles também não podem realizar tarefas usando o AWS Management Console, AWS Command Line Interface (AWS CLI), ou AWS API. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

Para saber como criar uma política IAM baseada em identidade usando esses exemplos de documentos de JSON política, consulte Criação de IAM políticas no Guia do IAMusuário.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Amazon EMR Serverless, incluindo o formato de cada um dos ARNs tipos de recursos, consulte Ações, recursos e chaves de condição do Amazon EMR Serverless na Referência de Autorização de Serviço.

Tópicos

- Melhores práticas de política
- Permitir que usuários visualizem suas próprias permissões

Melhores práticas de política

> **Note**
>
> EMRO Serverless não oferece suporte a políticas gerenciadas, portanto, a primeira prática listada abaixo não se aplica.

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon EMR Serverless em sua conta. Essas ações podem incorrer em custos para o seu Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com AWS políticas gerenciadas e migrar para permissões com privilégios mínimos — Para começar a conceder permissões para seus usuários e cargas de trabalho, use o AWS políticas gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis em seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo AWS políticas gerenciadas pelo cliente que são específicas para seus casos de uso. Para ter mais informações, consulte AWS políticas gerenciadas ou AWS políticas gerenciadas para funções de trabalho no Guia IAM do usuário.
- Aplique permissões com privilégios mínimos — Ao definir permissões com IAM políticas, conceda somente as permissões necessárias para realizar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre IAM como usar para aplicar permissões, consulte Políticas e permissões IAM no Guia IAM do usuário.
- Use condições nas IAM políticas para restringir ainda mais o acesso — Você pode adicionar uma condição às suas políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usandoSSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de um determinado AWS service (Serviço da AWS), por exemplo, AWS CloudFormation. Para obter mais informações, consulte Elementos IAM JSON da política: Condição no Guia IAM do usuário.
- Use o IAM Access Analyzer para validar suas IAM políticas e garantir permissões seguras e funcionais — o IAM Access Analyzer valida políticas novas e existentes para que as políticas sigam a linguagem da IAM política (JSON) e as melhores práticas. IAM IAMO Access Analyzer fornece mais de 100 verificações de políticas e recomendações práticas para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação da política do IAM Access Analyzer no Guia do IAM Usuário.
- Exigir autenticação multifatorial (MFA) — Se você tiver um cenário que exija IAM usuários ou um usuário root em seu Conta da AWS, ative MFA para obter segurança adicional. Para exigir MFA quando API as operações são chamadas, adicione MFA condições às suas políticas. Para obter mais informações, consulte Configurando o API acesso MFA protegido no Guia do IAMusuário.

Para obter mais informações sobre as melhores práticas emIAM, consulte as melhores práticas de segurança IAM no Guia IAM do usuário.

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permita IAM aos usuários visualizar as políticas embutidas e gerenciadas que estão anexadas à identidade do usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando o AWS CLI ou AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Atualizações do Amazon EMR Serverless para AWS políticas gerenciadas

Exibir detalhes sobre as atualizações do AWS gerenciei políticas para o Amazon EMR Serverless desde que esse serviço começou a rastrear essas mudanças. Para receber alertas automáticos sobre alterações nessa página, assine o RSS feed na página de histórico de documentos EMR sem servidor da Amazon.

Alteração	Descrição	Data
A mazonEMRServerless ServiceRolePolicy — Atualizaç ão de uma política existente	O Amazon EMR Serverles s adicionou o novo `Sid` `CloudWatchPolicySt atement` e `EC2Policy Statement` à política A. mazonEMRServerless ServiceRolePolicy	25 de janeiro de 2024
A mazonEMRServerless ServiceRolePolicy — Atualizaç ão de uma política existente	O Amazon EMR Serverless adicionou novas permissões para permitir que o Amazon EMR Serverless publique métricas de conta agregadas para uso de v CPU no namespace. `"AWS/Usage"`	20 de abril de 2023
A Amazon EMR Serverles s começou a monitorar as mudanças	A Amazon EMR Serverles s começou a monitorar as mudanças em seu AWS políticas gerenciadas.	20 de abril de 2023

Solução de problemas de identidade e acesso ao Amazon EMR Serverless

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon EMR Serverless e. IAM

Tópicos

- Não estou autorizado a realizar uma ação no Amazon EMR Serverless
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha AWS conta para acessar meus recursos Amazon EMR Serverless

Não estou autorizado a realizar uma ação no Amazon EMR Serverless

Se o AWS Management Console informa que você não está autorizado a realizar uma ação e, em seguida, entre em contato com o administrador para obter ajuda. O administrador é a pessoa que forneceu o seu nome de usuário e senha.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `emr-serverless:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas e permitir o acesso ao recurso *`my-example-widget`* usando a ação `emr-serverless:`*`GetWidget`*.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a `iam:PassRole` ação, suas políticas devem ser atualizadas para permitir que você passe uma função para a Amazon EMR Serverless.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um IAM usuário chamado `marymajor` tenta usar o console para realizar uma ação no Amazon EMR Serverless. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha AWS conta para acessar meus recursos Amazon EMR Serverless

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon EMR Serverless oferece suporte a esses recursos, consulte. Identity and Access Management (IAM) no Amazon EMR Serverless
- Para saber como fornecer acesso aos seus recursos em Contas da AWS que você possui, consulte Fornecendo acesso a um IAM usuário em outro Conta da AWS que você possui no Guia do IAM Usuário.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Fornecendo acesso a Contas da AWS propriedade de terceiros no Guia do IAM Usuário.
- Para saber como fornecer acesso por meio da federação de identidades, consulte Fornecendo acesso a usuários autenticados externamente (federação de identidades) no Guia do IAMusuário.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Usando EMR Serverless com AWS Lake Formation para controle de acesso refinado

Visão geral

Com as EMR versões 7.2.0 e superiores da Amazon, você pode aproveitar AWS Lake Formation para aplicar controles de acesso refinados em tabelas do Catálogo de Dados que são apoiadas pelo S3. Esse recurso permite configurar controles de acesso em nível de tabela, linha, coluna e célula para read consultas em suas tarefas do Amazon EMR Serverless Spark. Para configurar um controle

de acesso refinado para trabalhos em lote e sessões interativas do Apache Spark, use o Studio. EMR Consulte as seções a seguir para saber mais sobre o Lake Formation e como usá-lo com o EMR Serverless.

Usando o Amazon EMR Serverless com AWS Lake Formation incorre em custos adicionais. Para obter mais informações, consulte os EMRpreços da Amazon.

Como o EMR Serverless funciona com AWS Lake Formation

Usar o EMR Serverless com o Lake Formation permite aplicar uma camada de permissões em cada trabalho do Spark para aplicar o controle de permissões do Lake Formation quando EMR o Serverless executa trabalhos. EMRO Serverless usa perfis de recursos do Spark para criar dois perfis para executar tarefas com eficiência. O perfil do usuário executa o código fornecido pelo usuário, enquanto o perfil do sistema aplica as políticas do Lake Formation. Para obter mais informações, consulte O que é AWS Lake Formatione considerações e limitações.

Ao usar a capacidade pré-inicializada com o Lake Formation, recomendamos que você tenha no mínimo dois drivers Spark. Cada tarefa habilitada para Lake Formation utiliza dois drivers Spark, um para o perfil do usuário e outro para o perfil do sistema. Para obter o melhor desempenho, você deve usar o dobro do número de drivers para trabalhos habilitados para Lake Formation em comparação com aqueles que não usam Lake Formation.

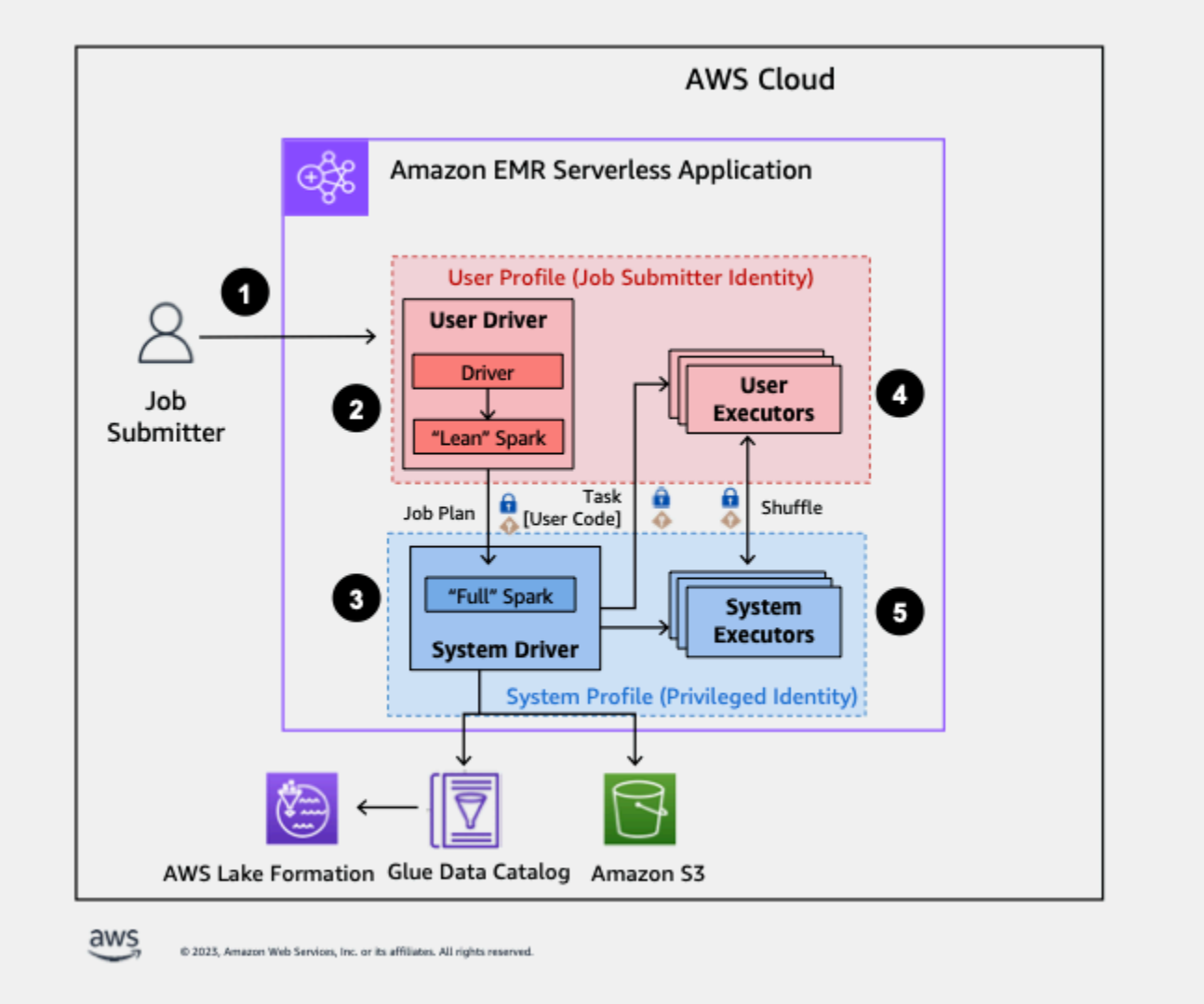
Ao executar trabalhos do Spark no EMR Serverless, você também deve considerar o impacto da alocação dinâmica no gerenciamento de recursos e no desempenho do cluster. A configuração `spark.dynamicAllocation.maxExecutors` do número máximo de executores por perfil de recurso se aplica aos executores do usuário e do sistema. Se você configurar esse número para ser igual ao número máximo permitido de executores, a execução do trabalho poderá travar devido a um tipo de executor que usa todos os recursos disponíveis, o que impede o outro executor ao executar trabalhos.

Para que você não fique sem recursos, o EMR Serverless define o número máximo padrão de executores por perfil de recurso como 90% do valor. `spark.dynamicAllocation.maxExecutors` Você pode substituir essa configuração ao especificar `spark.dynamicAllocation.maxExecutorsRatio` com um valor entre 0 e 1. Além disso, você também pode configurar as seguintes propriedades para otimizar a alocação de recursos e o desempenho geral:

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`

- `spark.cleaner.periodicGC.interval`

A seguir, uma visão geral de alto nível de como o EMR Serverless obtém acesso aos dados protegidos pelas políticas de segurança do Lake Formation.



1. Um usuário envia uma tarefa do Spark para um AWS Lake Formation- aplicativo sem EMR servidor habilitado.
2. EMRO Serverless envia o trabalho para um driver de usuário e executa o trabalho no perfil do usuário. O driver do usuário executa uma versão enxuta do Spark que não tem a capacidade de iniciar tarefas, solicitar executores, acessar o S3 ou o Glue Catalog. Ele cria um plano de trabalho.
3. EMRO Serverless configura um segundo driver chamado driver do sistema e o executa no perfil do sistema (com uma identidade privilegiada). EMRO Serverless configura um TLS

canal criptografado entre os dois drivers para comunicação. O driver do usuário usa o canal para enviar os planos de trabalho ao driver do sistema. O driver do sistema não executa o código enviado pelo usuário. Ele executa o Spark completo e se comunica com o S3 e com o Catálogo de Dados para acesso aos dados. Ele solicita executores e compila o Job Plan em uma sequência de estágios de execução.

4. EMRO Serverless então executa os estágios nos executores com o driver do usuário ou o driver do sistema. O código do usuário em qualquer estágio é executado exclusivamente nos executores do perfil do usuário.
5. Estágios que lêem dados de tabelas do Catálogo de Dados protegidas por AWS Lake Formation ou aqueles que aplicam filtros de segurança são delegados aos executores do sistema.

Possibilitando a formação de lagos na Amazônia EMR

Para habilitar o Lake Formation, você deve definir como `spark.emr-serverless.lakeformation.enabled` `true` `spark-defaults` subclassificação para o parâmetro de configuração de tempo de execução ao criar um aplicativo sem EMR servidor.

```
aws emr-serverless create-application \
    --release-label emr-7.2.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

Você também pode ativar o Lake Formation ao criar um novo aplicativo no EMR Studio. Escolha Use Lake Formation para um controle de acesso refinado, disponível em Configurações adicionais.

A criptografia entre trabalhadores é ativada por padrão quando você usa o Lake Formation com EMR Serverless, então você não precisa habilitar explicitamente a criptografia entre trabalhadores novamente.

Habilitando Lake Formation para trabalhos no Spark

Para ativar o Lake Formation para trabalhos individuais do Spark, `spark.emr-serverless.lakeformation.enabled` defina como true ao usar`spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

IAMPermissões da função Job Runtime

As permissões do Lake Formation controlam o acesso a AWS Cole os recursos do Data Catalog, os locais do Amazon S3 e os dados subjacentes nesses locais. IAMas permissões controlam o acesso à Lake Formation e AWS APIsGlue e recursos. Embora você possa ter a permissão do Lake Formation para acessar uma tabela no Catálogo de Dados (SELECT), sua operação falhará se você não tiver a IAM permissão na `glue:Get*` API operação.

A seguir está um exemplo de política de como fornecer IAM permissões para acessar um script no S3, fazendo o upload de registros para o S3, AWS Glue API as permissões e a permissão para acessar o Lake Formation.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScriptAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/scripts",
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/*" ]
        },
        {
         "Sid": "LoggingAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET/logs/*"
            ]
        },
        {
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "glue:Get*",
                "glue:Create*",
                "glue:Update*"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
}
```

Configurando permissões do Lake Formation para a função de tempo de execução do trabalho

Primeiro, registre a localização da sua mesa Hive no Lake Formation. Em seguida, crie permissões para sua função de tempo de execução de trabalho na tabela desejada. Para obter mais detalhes sobre Lake Formation, consulte O que é AWS Lake Formation? no AWS Lake Formation Guia do desenvolvedor.

Depois de configurar as permissões do Lake Formation, você pode enviar trabalhos do Spark no Amazon EMR Serverless. Para obter mais informações sobre trabalhos do Spark, consulte exemplos do Spark.

Enviando uma execução de trabalho

Depois de concluir a configuração dos subsídios do Lake Formation, você pode enviar trabalhos do Spark no EMR Serverless. Para executar trabalhos do Iceberg, você deve fornecer as seguintes `spark-submit` propriedades.

```
--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
```

Suporte ao formato de tabela aberta

A EMR versão 7.2.0 da Amazon inclui suporte para controle de acesso refinado com base no Lake Formation. EMRO Serverless é compatível com os tipos de tabela Hive e Iceberg. A tabela a seguir descreve todas as operações suportadas.

Operações	Hive	Iceberg
DDLcomandos	Somente com permissões de IAM função	Somente com permissões de IAM função
Consultas incrementais	Não aplicável	Suporte total
Consultas de viagem no tempo	Não aplicável a esse formato de tabela	Suporte total
Tabelas de metadados	Não aplicável a esse formato de tabela	Compatível, mas algumas tabelas estão ocultas. Consulte considerações e limitações para obter mais informações.
`DML INSERT`	Somente com IAM permissões	Somente com IAM permissões
DML UPDATE	Não aplicável a esse formato de tabela	Somente com IAM permissões
`DML DELETE`	Não aplicável a esse formato de tabela	Somente com IAM permissões
Operações de leitura	Suporte total	Suporte total
Procedimentos armazenados	Não aplicável	Compatível com as exceções de `register_table` e. `migrate` Consulte considera

Operações	Hive	Iceberg
		ções e limitações para obter mais informações.

Considerações e limitações

Considere as seguintes considerações e limitações ao usar o Lake Formation com EMR Serverless.

> **Note**
>
> Quando você ativa o Lake Formation para uma tarefa do Spark no EMR Serverless, a tarefa inicia um driver de sistema e um driver de usuário. Se você especificou a capacidade pré-inicializada na inicialização, os drivers são provisionados a partir da capacidade pré-inicializada e o número de drivers do sistema é igual ao número de drivers de usuário que você especifica. Se você escolher a capacidade sob demanda, o EMR Serverless iniciará um driver de sistema além de um driver de usuário. Para estimar os custos associados ao seu trabalho EMR Serverless with Lake Formation, use o AWS Pricing Calculator.

O Amazon EMR Serverless com Lake Formation está disponível em todas as regiões sem EMRservidor suportadas, exceto AWS GovCloud (Leste dos EUA) e AWS GovCloud (Oeste dos EUA).

- O Amazon EMR Serverless oferece suporte ao controle de acesso refinado via Lake Formation somente para tabelas Apache Hive e Apache Iceberg. Os formatos do Apache Hive incluem Parquet e xSvORC.
- Os aplicativos habilitados para Lake Formation não oferecem suporte ao uso de imagens personalizadas EMR sem servidor.
- Você não pode se `DynamicResourceAllocation` candidatar a empregos em Lake Formation.
- Você só pode usar o Lake Formation com tarefas do Spark.
- EMRO Serverless with Lake Formation suporta apenas uma única sessão do Spark durante um trabalho.
- EMRO Serverless with Lake Formation suporta apenas consultas de tabelas entre contas compartilhadas por meio de links de recursos.
- O seguinte não é suportado:

  - Conjuntos de dados distribuídos resilientes () RDD
  - Streaming do Spark
  - Escreva com as permissões concedidas pelo Lake Formation
  - Controle de acesso para colunas aninhadas
- EMRO servidor bloqueia funcionalidades que podem prejudicar o isolamento completo do driver do sistema, incluindo as seguintes:
  - UDTsiveUDFs, H e qualquer função definida pelo usuário que envolva classes personalizadas
  - Fontes de dados personalizadas
  - Fornecimento de frascos adicionais para extensão, conector ou metastore do Spark
  - `ANALYZE TABLE` command
- Para impor controles de acesso `EXPLAIN PLAN` e DDL operações como `DESCRIBE TABLE` não expor informações restritas.
- EMRO servidor sem servidor restringe o acesso aos registros do Spark do driver do sistema em aplicativos habilitados para Lake Formation. Como o driver do sistema é executado com mais acesso, os eventos e registros que o driver do sistema gera podem incluir informações confidenciais. Para impedir que usuários ou códigos não autorizados acessem esses dados confidenciais, o EMR Serverless desativou o acesso aos registros do driver do sistema. Para solução de problemas, entre em contato AWS apoio.
- Se você registrou uma localização de tabela no Lake Formation, o caminho de acesso aos dados passa pelas credenciais armazenadas do Lake Formation, independentemente da IAM permissão para a função de tempo de execução do trabalho EMR sem servidor. Se você configurar incorretamente a função registrada com a localização da tabela, os trabalhos enviados que usam a função com IAM permissão do S3 para a localização da tabela falharão.
- Escrever em uma tabela do Lake Formation usa a IAM permissão em vez das permissões concedidas pelo Lake Formation. Se sua função de tempo de execução de trabalho tiver as permissões necessárias do S3, você poderá usá-la para executar operações de gravação.

A seguir estão as considerações e limitações ao usar o Apache Iceberg:

- Você só pode usar o Apache Iceberg com o catálogo de sessões e não com catálogos nomeados arbitrariamente.
- As tabelas Iceberg registradas no Lake Formation oferecem suporte apenas às tabelas de metadados `history`metadata_log_entries`,,snapshots`, `filesmanifests`, e. `refs` A Amazon EMR oculta as colunas que podem conter dados confidenciais`partitions`, como`path`,

e. `summaries` Essa limitação não se aplica às tabelas Iceberg que não estão registradas no Lake Formation.

- As tabelas que você não registra no Lake Formation oferecem suporte a todos os procedimentos armazenados do Iceberg. Os `migrate` procedimentos `register_table` e não são compatíveis com nenhuma tabela.
- Recomendamos que você use o Iceberg DataFrameWriter V2 em vez do V1.

Solução de problemas

Consulte as seções a seguir para obter soluções de problemas.

Registro em log

EMRO Serverless usa perfis de recursos do Spark para dividir a execução do trabalho. EMRO Serverless usa o perfil do usuário para executar o código que você forneceu, enquanto o perfil do sistema aplica as políticas do Lake Formation. Você pode acessar os registros das tarefas executadas como perfil de usuário.

UI ao vivo e servidor de histórico do Spark

A Live UI e o Spark History Server têm todos os eventos do Spark gerados a partir do perfil do usuário e os eventos editados gerados pelo driver do sistema.

Você pode ver todas as tarefas dos drivers do usuário e do sistema na guia Executors. No entanto, os links de registro estão disponíveis somente para o perfil do usuário. Além disso, algumas informações são retiradas da Live UI, como o número de registros de saída.

O trabalho falhou com permissões insuficientes do Lake Formation

Certifique-se de que sua função de tempo de execução de trabalho tenha as permissões para ser executada SELECT e estar DESCRIBE na tabela que você está acessando.

Job com RDD falha na execução

EMRAtualmente, o Serverless não oferece suporte a operações resilientes de conjunto de dados distribuído (RDD) em trabalhos habilitados para Lake Formation.

Não é possível acessar arquivos de dados no Amazon S3

Certifique-se de ter registrado a localização do data lake em Lake Formation.

Exceção de validação de

EMRO Serverless detectou um erro de validação de segurança. Contato AWS suporte para assistência.

Compartilhamento AWS Glue o catálogo de dados e as tabelas em todas as contas

Você pode compartilhar bancos de dados e tabelas entre contas e ainda usar o Lake Formation. Para obter mais informações, consulte Compartilhamento de dados entre contas no Lake Formation e Como faço para compartilhar AWS Glue Data Catalog e tabelas em várias contas usando AWS Lake Formation?.

Criptografia entre trabalhadores

Com EMR as versões 6.15.0 e superiores da Amazon, você pode habilitar a comunicação TLS criptografada mútua entre trabalhadores em suas execuções de trabalho do Spark. Quando ativado, o EMR Serverless gera e distribui automaticamente um certificado exclusivo para cada trabalhador provisionado em suas execuções de trabalho. Quando esses trabalhadores se comunicam para trocar mensagens de controle ou transferir dados aleatórios, eles estabelecem uma TLS conexão mútua e usam os certificados configurados para verificar a identidade uns dos outros. Se um trabalhador não conseguir verificar outro certificado, o TLS handshake falhará e o EMR Serverless abortará a conexão entre eles.

Se você estiver usando o Lake Formation com EMR Serverless, a TLS criptografia mútua é ativada por padrão.

Habilitando a TLS criptografia mútua no EMR Serverless

Para ativar a TLS criptografia mútua em seu aplicativo Spark, `spark.ssl.internode.enabled` defina como true ao criar um aplicativo EMR sem servidor. Se você estiver usando o AWS console para criar um aplicativo EMR sem servidor, escolha Usar configurações personalizadas, expanda Configuração do aplicativo e insira seu. `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"
```

Se você quiser ativar a TLS criptografia mútua para execuções individuais de tarefas do Spark, `spark.ssl.internode.enabled` defina como true ao usar`spark-submit`.

```
--conf spark.ssl.internode.enabled=true
```

Secrets Manager para proteção de dados com EMR Serverless

AWS Secrets Manager é um serviço de armazenamento secreto que você pode usar para proteger credenciais, API chaves e outras informações secretas do banco de dados. Em seguida, em seu código, você pode substituir as credenciais codificadas por uma API chamada para o Secrets Manager. Isso ajuda a garantir que o segredo não possa ser comprometido por alguém examinando seu código, porque o segredo não está lá. Para obter uma visão geral, consulte o AWS Secrets Manager Guia do usuário.

Secrets Manager criptografa segredos usando AWS Key Management Service chaves. Para obter mais informações, consulte Criptografia e decodificação secretas no AWS Secrets Manager Guia do usuário.

Você pode configurar o Secrets Manager para alterar automaticamente os segredos para você de acordo com a programação que você especificar. Isso permite substituir segredos de longo prazo por outros de curto prazo, ajudando a reduzir de maneira significativa o risco de comprometimento. Para obter mais informações, consulte Girar AWS Secrets Manager segredos no AWS Secrets Manager Guia do usuário.

O Amazon EMR Serverless se integra com AWS Secrets Manager para que você possa armazenar seus dados no Secrets Manager e usar o ID secreto em suas configurações.

Como o EMR Serverless usa segredos

Quando você armazena seus dados no Secrets Manager e usa o ID secreto em suas configurações para o EMR Serverless, você não passa dados de configuração confidenciais para o EMR Serverless em texto simples e os expõe externamente. APIs Se você indicar que um par de valores-chave contém uma ID secreta para um segredo que você armazenou no Secrets Manager, o EMR Serverless recuperará o segredo ao enviar dados de configuração aos trabalhadores para execução de trabalhos.

Para indicar que um par de valores-chave de uma configuração contém uma referência a um segredo armazenado no Secrets Manager, adicione a `EMR.secret@` anotação ao valor da configuração. Para qualquer propriedade de configuração com anotação de ID secreta, o EMR Serverless chama o Secrets Manager e resolve o segredo no momento da execução do trabalho.

Como criar um segredo

Para criar um segredo, siga as etapas em Criar um AWS Secrets Manager segredo no AWS Secrets Manager Guia do usuário. Na Etapa 3, escolha o campo Texto simples para inserir seu valor confidencial.

Forneça um segredo em uma classificação de configuração

Os exemplos a seguir mostram como fornecer um segredo em uma classificação de configuração em`StartJobRun`. Se você quiser configurar classificações para o Secrets Manager no nível do aplicativo, consulteConfiguração padrão do aplicativo para EMR Serverless.

Nos exemplos, *`SecretName`* substitua pelo nome do segredo a ser recuperado. Inclua o hífen, seguido pelos seis caracteres que o Secrets Manager adiciona ao final do segredoARN. Para obter mais informações, consulte Como criar um segredo.

Nesta seção

Especifique referências secretas - Spark

Example — Especifique referências secretas na configuração externa do metastore Hive para o Spark

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

Example — Especifique referências secretas para a configuração externa do metastore Hive na classificação **`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",
        }
    }
```

Especifique referências secretas - Hive

Example — Especifique referências secretas na configuração externa do metastore Hive para o Hive

```
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.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-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://EXAMPLE-LOG-BUCKET"
        }
    }
}'
```

Example — Especifique referências secretas para a configuração externa do metastore Hive na classificação **`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"
    }
}
```

Conceda acesso ao EMR Serverless para recuperar o segredo

Para permitir que o EMR Serverless recupere o valor secreto do Secrets Manager, adicione a seguinte declaração de política ao seu segredo ao criá-lo. Você deve criar seu segredo com a KMS chave gerenciada pelo cliente para que o EMR Serverless leia o valor secreto. Para obter mais informações, consulte Permissões para a KMS chave no AWS Secrets Manager Guia do usuário.

Na política a seguir, *applicationId* substitua pela ID do seu aplicativo.

Política de recursos para o segredo

Você deve incluir as seguintes permissões na política de recursos para o segredo em AWS Secrets Manager para permitir que o EMR Serverless recupere valores secretos. Para garantir que somente um aplicativo específico possa recuperar esse segredo, você pode, opcionalmente, especificar o ID do aplicativo EMR sem servidor como uma condição na política.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:Região da AWS:aws_account_id:/
applications/applicationId"
        }
      }
    }
  ]
}
```

Crie seu segredo com a seguinte política para pessoas gerenciadas pelo cliente AWS Key Management Service (AWS KMS) chave:

Política para gerenciamento de clientes AWS KMS chave

```
{
    "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.Região da AWS.amazonaws.com"
        }
    }
}
```

Girando o segredo

A rotação é quando você atualiza periodicamente um segredo. Você pode configurar AWS Secrets Manager para alternar automaticamente o segredo para você em uma programação especificada por você. Dessa forma, você pode substituir segredos de longo prazo por segredos de curto prazo. Isso ajuda a reduzir o risco de comprometimento. EMRO Serverless recupera o valor secreto de uma configuração anotada quando a tarefa passa para um estado em execução. Se você ou um processo atualizar o valor secreto no Secrets Manager, você deverá enviar um novo trabalho para que o trabalho possa buscar o valor atualizado.

> **Note**
>
> Os trabalhos que já estão em execução não podem obter um valor secreto atualizado. Isso pode resultar em falha no trabalho.

Usando as concessões de acesso do Amazon S3 com o Serverless EMR

Visão geral do S3 Access Grants para EMR servidores sem servidor

Com as EMR versões 6.15.0 e superiores da Amazon, o Amazon S3 Access Grants fornece uma solução de controle de acesso escalável que você pode usar para aumentar o acesso aos seus dados do Amazon S3 a partir do Serverless. EMR Se você tiver uma configuração de permissão complexa ou grande para os dados do S3, poderá usar a funcionalidade Access Grants para escalar as permissões de dados do S3 para usuários, perfis e aplicações.

Use o S3 Access Grants para aumentar o acesso aos dados do Amazon S3 além das permissões concedidas pela função de tempo de execução ou IAM pelas funções anexadas às identidades com acesso ao seu aplicativo sem servidor. EMR

Para obter mais informações, consulte Gerenciamento de acesso com concessões de acesso do S3 para a Amazon EMR no Guia de gerenciamento da Amazon e EMR Gerenciamento do acesso com concessões de acesso do S3 no Guia do usuário do Amazon Simple Storage Service.

Esta seção descreve como iniciar um aplicativo EMR sem servidor que usa o S3 Access Grants para fornecer acesso aos dados no Amazon S3. Para ver as etapas de uso do S3 Access Grants com outras EMR implantações da Amazon, consulte a seguinte documentação:

- Usando o S3 Access Grants com a Amazon EMR
- Usando o S3 Access Grants com a Amazon EMR em EKS

Inicie um aplicativo EMR sem servidor com o S3 Access Grants para gerenciamento de dados

Você pode ativar o S3 Access Grants no EMR Serverless e iniciar um aplicativo Spark. Quando sua aplicação solicita dados do S3, o Amazon S3 fornece credenciais temporárias que têm como escopo o bucket, prefixo ou objeto específico.

1. Configure uma função de execução de tarefas para seu EMR aplicativo sem servidor. Inclua IAM as permissões necessárias para executar trabalhos do Spark e usar o S3 Access Grants e: `s3:GetDataAccess` `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**
>
> Se você especificar IAM funções para execução de trabalhos que tenham permissões adicionais para acessar o S3 diretamente, os usuários poderão acessar os dados permitidos pela função mesmo que não tenham permissão do S3 Access Grants.

2. Inicie seu aplicativo EMR Serverless com uma etiqueta de EMR lançamento da Amazon de 6.15.0 ou superior e a `spark-defaults` classificação, conforme mostra o exemplo a seguir. Substitua os valores em *`red text`* pelos valores apropriados ao seu cenário de uso.

```
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://DOC-EXAMPLE-BUCKET-OUTPUT/
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"
         }
```

```
      }]
}'
```

O S3 Access concede considerações com o Serverless EMR

Para obter informações importantes sobre suporte, compatibilidade e comportamento ao usar o Amazon S3 Access Grants com EMR Serverless, consulte Considerações sobre o S3 Access Grants com a Amazon no EMR Amazon Management Guide. EMR

Registrando chamadas Amazon EMR Serverless usando API AWS CloudTrail

O Amazon EMR Serverless é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um AWS serviço em EMR Serverless. CloudTrail captura todas as API chamadas para EMR Serverless como eventos. As chamadas capturadas incluem chamadas do console EMR Serverless e chamadas de código para as operações EMR API Serverless. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para EMR Serverless. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita à EMR Serverless, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre CloudTrail, consulte o AWS CloudTrail Guia do usuário.

EMRInformações sem servidor em CloudTrail

CloudTrail está habilitado em seu Conta da AWS quando você cria a conta. Quando a atividade ocorre no EMR Serverless, essa atividade é registrada em um CloudTrail evento junto com outros AWS eventos de serviço no Histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em seu Conta da AWS. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em seu Conta da AWS, incluindo eventos para EMR Serverless, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, a trilha se aplica a todos Regiões da AWS. A trilha registra eventos de todas as regiões do AWS particiona e entrega os arquivos de

log para o bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir ainda mais com base nos dados do evento coletados nos CloudTrail registros. Para obter mais informações, consulte as informações a seguir.

- Visão geral da criação de uma trilha
- CloudTrail serviços e integrações suportados
- Configurando SNS notificações da Amazon para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e Recebendo arquivos de CloudTrail log de várias contas

Todas as ações EMR sem servidor são registradas CloudTrail e documentadas na Referência sem servidor. EMR API Por exemplo, chamadas para o `CreateApplication` `StartJobRun` e `CancelJobRun` as ações geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com root ou AWS Identity and Access Management (IAM) credenciais do usuário.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte o CloudTrail userIdentity elemento.

Entendendo as entradas do arquivo de log EMR sem servidor

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das API chamadas públicas, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a `CreateApplication` ação.

```
{
    "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"
}
```

Validação de conformidade para Amazon EMR Serverless

A segurança e a conformidade do EMR Serverless são avaliadas por auditores terceirizados como parte de vários AWS programas de conformidade, incluindo os seguintes:

- Controles do sistema e da organização (SOC)
- Padrão de segurança de dados do setor de cartões de pagamento (PCIDSS)
- Programa Federal de Gestão de Riscos e Autorizações (FedRAMP) Moderado
- Lei de Portabilidade e Responsabilidade de Seguros de Saúde () HIPAA

AWS fornece uma lista frequentemente atualizada de AWS serviços no escopo de programas de conformidade específicos em AWS Serviços no escopo do Programa de Conformidade.

Relatórios de auditoria de terceiros estão disponíveis para você baixar usando AWS Artifact. Para obter mais informações, consulte Baixando relatórios em AWS Artifato.

Para obter mais informações sobre AWS programas de conformidade, consulte AWS Programas de conformidade.

Sua responsabilidade de conformidade ao usar o EMR Serverless é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua organização e pelas leis e regulamentos aplicáveis. Se o uso do EMR Serverless estiver sujeito à conformidade com padrões comoHIPAA,, ou Fed PCI RAMP Moderate, AWS fornece recursos para ajudar a:

- Guias de início rápido sobre segurança e conformidade que discutem considerações arquitetônicas e etapas para a implantação de ambientes básicos focados em segurança e conformidade em AWS.
- AWS Os guias de conformidade do cliente podem ajudá-lo a entender o modelo de responsabilidade compartilhada sob o prisma da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeie a orientação para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de

Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).

- AWS Config O pode ser usado para avaliar até que ponto suas configurações de recursos atendem adequadamente a práticas internas e a diretrizes e regulamentações do setor.
- AWS Os Recursos de Conformidade são uma coleção de pastas de trabalho e guias que podem ser aplicados ao seu setor e localização.
- AWS O Security Hub fornece uma visão abrangente do seu estado de segurança em AWS e ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.
- AWS Audit Manager— isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon Serverless EMR

A ferramenta AWS a infraestrutura global é construída em torno de AWS Regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, conectadas com baixa latência, throughput elevado e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenters tradicionais.

Para obter mais informações sobre AWS Regiões e zonas de disponibilidade, consulte AWS Infraestrutura global.

Além do AWS infraestrutura global, o Amazon EMR Serverless oferece integração com o Amazon S3 para ajudar a suportar suas EMRFS necessidades de resiliência e backup de dados.

Segurança da infraestrutura no Amazon EMR Serverless

Como um serviço gerenciado, a Amazon EMR é protegida por AWS segurança de rede global. Para obter mais informações sobre AWS serviços de segurança e como AWS protege a infraestrutura, consulte AWS Segurança na nuvem. Para projetar seu AWS ambiente usando as melhores práticas para segurança de infraestrutura, consulte Proteção de infraestrutura no pilar de segurança AWS Estrutura bem arquitetada.

Você usa AWS APIchamadas publicadas para acessar a Amazon EMR pela rede. Os clientes devem oferecer suporte para:

- Segurança da camada de transporte (TLS). Exigimos TLS 1,2 e recomendamos TLS 1,3.
- Suítes de criptografia com sigilo direto perfeito (), como (Ephemeral PFS Diffie-Hellman) ou DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando uma ID de chave de acesso e uma chave de acesso secreta associada a um IAM principal. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Análise de configuração e vulnerabilidade no Amazon EMR Serverless

AWS lida com tarefas básicas de segurança, como sistema operacional (SO) convidado e aplicação de patches em bancos de dados, configuração de firewall e recuperação de desastres. Esses procedimentos foram revisados e certificados por terceiros certificados. Para obter mais detalhes, consulte os seguintes recursos da :

- Validação de conformidade para Amazon EMR Serverless
- Modelo de responsabilidade compartilhada
- Amazon Web Services: visão geral do processo de segurança (whitepaper)

Endpoints e cotas para EMR Serverless

Service endpoints

Para se conectar programaticamente a um AWS service (Serviço da AWS), você usa um endpoint. Um endpoint é o ponto URL de entrada para um AWS serviço web. Além do padrão AWS endpoints, alguns Serviços da AWS oferecem FIPS endpoints em regiões selecionadas. A tabela a seguir lista os endpoints de serviço do EMR Serverless. Para ter mais informações, consulte AWS service (Serviço da AWS) endpoints.

Nome da região	Região	Endpoint	Protocolo
Leste dos EUA (Ohio)	`us-east-2` (limitado às seguintes zonas de disponibilidade:`use2-az1`,`use2-az2`, e`use2-az3`)	`emr-serverless.us-east-2.amazonaws.com`	HTTPS

Leste dos EUA (Norte da Virgínia)	`us-east-1` (limitado às seguintes zonas de disponibilidade: `use1-az1`,`use1-az2`,`use1-az4`,`use1-az5`, e`use1-az6`)	`emr-serverless.us-east-1.amazonaws.com`
		
		`emr-serverless-fips.us-east-1.amazonaws.com`
			HTTPS

Oeste dos EUA (N. da Califórnia)	`us-west-1`	`emr-serverless.us-west-1.amazonaws.com`	HTTPS
Oeste dos EUA (Oregon)	`us-west-2`	`emr-serverless.us-west-2.amazonaws.com`	HTTPS

Nome da região	Região	Endpoint	Protocolo
		emr-serverless-fips.us-west-2.amazonaws.com	
África (Cidade do Cabo)	af-south-1	emr-serverless.af-south-1.amazonaws.com	HTTPS
Ásia-Pacífico (Hong Kong)	ap-east-1	emr-serverless.ap-east-1.amazonaws.com	HTTPS
Ásia-Pacífico (Jacarta)	ap-southeast-3	emr-serverless.ap-southeast-3.amazonaws.com	HTTPS
Ásia-Pacífico (Mumbai)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
Ásia-Pacífico (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS

Nome da região	Região	Endpoint	Protocolo
Ásia-Pacífico (Seul)	`ap-northeast-2`	`emr-serverless.ap-northeast-2.amazonaws.com`	HTTPS
Ásia-Pacífico (Singapura)	`ap-southeast-1`	`emr-serverless.ap-southeast-1.amazonaws.com`	HTTPS
Ásia-Pacífico (Sydney)	`ap-southeast-2`	`emr-serverless.ap-southeast-2.amazonaws.com`	HTTPS
Ásia-Pacífico (Tóquio)	`ap-northeast-1`	`emr-serverless.ap-northeast-1.amazonaws.com`	HTTPS
Canadá (Central)	`ca-central-1` (limitado às seguintes zonas de disponibilidade: `cac1-az1` `ecac1-az2`)	`emr-serverless.ca-central-1.amazonaws.com`	HTTPS
Europa (Frankfurt)	`eu-central-1`	`emr-serverless.eu-central-1.amazonaws.com`	HTTPS

Nome da região	Região	Endpoint	Protocolo
Europa (Irlanda)	`eu-west-1`	`emr-serverless.eu-west-1.amazonaws.com`	HTTPS
Europa (Londres)	`eu-west-2`	`emr-serverless.eu-west-2.amazonaws.com`	HTTPS
Europa (Milão)	`eu-south-1`	`emr-serverless.eu-south-1.amazonaws.com`	HTTPS
Europe (Paris)	`eu-west-3`	`emr-serverless.eu-west-3.amazonaws.com`	HTTPS
Europa (Espanha)	`eu-south-2`	`emr-serverless.eu-south-2.amazonaws.com`	HTTPS
Europe (Stockholm)	`eu-north-1`	`emr-serverless.eu-north-1.amazonaws.com`	HTTPS
Oriente Médio (Barém)	`me-south-1`	`emr-serverless.me-south-1.amazonaws.com`	HTTPS

Nome da região	Região	Endpoint	Protocolo
Oriente Médio (UAE)	`me-central-1`	`emr-serverless.me-central-1.amazonaws.com`	HTTPS
América do Sul (São Paulo)	`sa-east-1`	`emr-serverless.sa-east-1.amazonaws.com`	HTTPS
AWS GovCloud (Leste dos EUA)	`us-gov-east-1`	`emr-serverless.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (Oeste dos EUA)	`us-gov-west-1`	`emr-serverless.us-gov-west-1.amazonaws.com`	HTTPS

Cotas de serviço

As cotas de serviço, também conhecidas como limites, são o número máximo de recursos ou operações de serviço que seu Conta da AWS pode usar. EMRO Serverless coleta métricas de uso da cota de serviço a cada minuto e as publica no namespace. `AWS/Usage`

> **Note**
>
> Novo AWS as contas podem ter cotas iniciais mais baixas que podem aumentar com o tempo. O Amazon EMR Serverless monitora o uso da conta em cada Região da AWS e, em seguida, aumenta automaticamente as cotas com base no seu uso.

A tabela a seguir lista as cotas de serviço para EMR Serverless. Para ter mais informações, consulte AWS service (Serviço da AWS) cotas.

Nome	Limite padrão	Ajustável?	Descrição
Máximo de simultâne os vCPUs por conta	16	Sim	O número máximo vCPUs que pode ser executado simultane amente para a conta atual Região da AWS.

APIlimites

A seguir, descrevemos os API limites por região para seu Conta da AWS.

Recurso	Cota padrão
ListApplications	10 transações por segundo. Explosão de 50 transações por segundo.
CreateApplication	1 transação por segundo. Explosão de 25 transações por segundo.
DeleteApplication	1 transação por segundo. Explosão de 25 transações por segundo.
GetApplication	10 transações por segundo. Explosão de 50 transações por segundo.
UpdateApplication	1 transação por segundo. Explosão de 25 transações por segundo.
ListJobRuns	1 transação por segundo. Explosão de 25 transações por segundo.
StartJobRun	1 transação por segundo. Explosão de 25 transações por segundo.
GetDashboardForJobRun	1 transação por segundo. Explosão de 2 transações por segundo.

Recurso	Cota padrão
CancelJobRun	1 transação por segundo. Explosão de 25 transações por segundo.
GetJobRun	10 transações por segundo. Explosão de 50 transações por segundo.
StartApplication	1 transação por segundo. Explosão de 25 transações por segundo.
StopApplication	1 transação por segundo. Explosão de 25 transações por segundo.

Outras considerações

A lista a seguir contém outras considerações sobre o EMR Serverless.

- EMRO Serverless está disponível no seguinte Regiões da AWS:
  - Leste dos EUA (Ohio)
  - Leste dos EUA (N. da Virgínia)
  - Oeste dos EUA (N. da Califórnia)
  - Oeste dos EUA (Oregon)
  - África (Cidade do Cabo)
  - Ásia-Pacífico (Hong Kong)
  - Ásia-Pacífico (Jacarta)
  - Ásia-Pacífico (Mumbai)
  - Ásia-Pacífico (Osaka)
  - Ásia-Pacífico (Seul)
  - Ásia-Pacífico (Singapura)
  - Ásia-Pacífico (Sydney)
  - Ásia-Pacífico (Tóquio)
  - Canadá (Central)
  - Europa (Frankfurt)
  - Europa (Irlanda)
  - Europa (Londres)
  - Europa (Milão)
  - Europe (Paris)
  - Europa (Espanha)
  - Europe (Stockholm)
  - Oriente Médio (Barém)
  - Oriente Médio (UAE)
  - América do Sul (São Paulo)
  - AWS GovCloud (Leste dos EUA)
  - AWS GovCloud (Oeste dos EUA)

Para obter uma lista dos endpoints associados a essas regiões, consulteService endpoints.

- O tempo limite padrão para a execução de um trabalho é de 12 horas. Você pode alterar essa configuração com a `executionTimeoutMinutes` propriedade no `startJobRun` API ou no AWS SDK. Você pode definir como 0 `executionTimeoutMinutes` se quiser que a execução do trabalho nunca atinja o tempo limite. Por exemplo, se você tiver um aplicativo de streaming, poderá definir como 0 `executionTimeoutMinutes` para permitir que a tarefa de streaming seja executada continuamente.
- A `billedResourceUtilization` propriedade no `getJobRun` API mostra o agregado vCPU, a memória e o armazenamento que AWS cobrou pela execução do trabalho. Os recursos cobrados incluem um uso mínimo de 1 minuto para funcionários, além de armazenamento adicional de mais de 20 GB por funcionário. Esses recursos não incluem o uso de trabalhadores pré-inicializados ociosos.
- Sem VPC conectividade, um trabalho pode acessar alguns AWS service (Serviço da AWS) endpoints no mesmo Região da AWS. Esses serviços incluem o Amazon S3, AWS Glue, Amazon CloudWatch Logs, AWS KMS, AWS Security Token Service, Amazon DynamoDB e AWS Secrets Manager. Você pode ativar a VPC conectividade para acessar outros Serviços da AWS através AWS PrivateLink, mas você não é obrigado a fazer isso. Para acessar serviços externos, você pode criar seu aplicativo com umVPC.
- EMRO Serverless não oferece suporte. HDFS Os discos locais dos trabalhadores são armazenamento temporal que o EMR Serverless usa para embaralhar e processar dados durante a execução do trabalho.
- EMRO Serverless não oferece suporte ao existente. emr-dynamodb-connector

Versões de EMR lançamento do Amazon Serverless

Uma EMR versão da Amazon é um conjunto de aplicativos de código aberto do ecossistema de big data. Cada versão inclui aplicativos, componentes e recursos de big data que você seleciona para que o Amazon EMR Serverless implante e configure ao executar seu trabalho.

Com o Amazon EMR 6.6.0 e superior, você pode implantar EMR o Serverless. Essa opção de implantação não está disponível nas EMR versões anteriores da Amazon. Ao enviar seu trabalho, você deve especificar uma das seguintes versões compatíveis.

Tópicos

EMR Serverless 7.2.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 7.2.0.

Aplicativo	Version (Versão)
Apache Spark	3.5.1

Aplicativo	Version (Versão)
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de lançamento do Serverless 7.2.0

- Lake Formation com EMR Serverless — agora você pode usar AWS Lake Formation para aplicar controles de acesso refinados em tabelas do Catálogo de Dados que são apoiadas pelo S3. Esse recurso permite que você configure controles de acesso em nível de tabela, linha, coluna e célula para consultas de leitura em suas tarefas do EMR Serverless Spark. Para ter mais informações, consulte the section called “Lake Formation para FGAC” e the section called “Considerações”.

EMR Serverless 7.1.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 7.1.0.

Aplicativo	Version (Versão)
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.0.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 7.0.0.

Aplicativo	Version (Versão)
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.15.0.

Aplicativo	Version (Versão)
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de lançamento do Serverless 6.15.0

- TLSsuporte — Com as versões 6.15.0 e posteriores do Amazon EMR Serverless, você pode habilitar a comunicação TLS criptografada mútua entre os trabalhadores em suas execuções de trabalho do Spark. Quando ativado, o EMR Serverless gera automaticamente um certificado exclusivo para cada funcionário, provisionado em uma execução de trabalho que os trabalhadores utilizam durante o TLS handshake para se autenticarem e estabelecerem um canal criptografado para processar dados com segurança. Para obter mais informações sobre TLS criptografia mútua, consulte Criptografia entre trabalhadores.

EMR Serverless 6.14.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.14.0.

Aplicativo	Version (Versão)
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.13.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.13.0.

Aplicativo	Version (Versão)
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.12.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.12.0.

Aplicativo	Version (Versão)
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.11.0.

Aplicativo	Version (Versão)
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de lançamento do Serverless 6.11.0

- Acesse recursos do S3 em outras contas - Com as versões 6.11.0 e superiores, você pode configurar várias IAM funções a serem assumidas ao acessar buckets do Amazon S3 em diferentes AWS contas da EMR Serverless.

EMR Serverless 6.10.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.10.0.

Aplicativo	Version (Versão)
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de lançamento do Serverless 6.10.0

- Para aplicativos EMR sem servidor com versão 6.10.0 ou superior, o valor padrão da propriedade é. `spark.dynamicAllocation.maxExecutors` `infinity` As versões anteriores são padronizadas para`100`. Para obter mais informações, consulte Propriedades de trabalho no Spark.

EMR Serverless 6.9.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.9.0.

Aplicativo	Version (Versão)
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de lançamento do Serverless 6.9.0

- A integração do Amazon Redshift para o Apache Spark está incluída nas EMR versões 6.9.0 e posteriores da Amazon. Anteriormente uma ferramenta de código aberto, a integração nativa é um conector do Spark que você pode usar para criar aplicações do Apache Spark que realizam a leitura e a gravação de dados no Amazon Redshift e no Amazon Redshift sem servidor. Para obter

mais informações, consulte Usando a integração do Amazon Redshift para o Apache Spark no Amazon Serverless EMR.

- EMRA versão sem servidor 6.9.0 adiciona suporte para AWS Arquitetura Graviton2 (arm64). Você pode usar o `architecture` parâmetro para `create-application` e `update-application` APIs para escolher a arquitetura arm64. Para obter mais informações, consulte Opções de EMR arquitetura Amazon Serverless.
- Agora você pode exportar, importar, consultar e unir tabelas do Amazon DynamoDB diretamente dos EMR seus aplicativos Serverless Spark e Hive. Para obter mais informações, consulte Conectando-se ao DynamoDB com o Amazon Serverless EMR.

Problemas conhecidos

- Se você usar a integração do Amazon Redshift para Apache Spark e tiver um time, timetz, timestamp ou timestamptz com precisão de microssegundos no formato Parquet, o conector arredondará os valores de tempo para o valor de milissegundo mais próximo. Como solução alternativa, use o parâmetro `unload_s3_format` do formato de descarregamento de texto.

EMR Serverless 6.8.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.8.0.

Aplicativo	Version (Versão)
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.7.0.

Aplicativo	Version (Versão)
Apache Spark	3.2.1

Aplicativo	Version (Versão)
Apache Hive	3.1.3
Apache Tez	0.9.2

Alterações, aprimoramentos e problemas resolvidos específicos do motor

A tabela a seguir lista um novo recurso específico do mecanismo.

Alteração	Descrição
Atributo	O agendador Tez agora suporta a preempção da tarefa Tez em vez da preempção do contêiner

EMR Serverless 6.6.0

A tabela a seguir lista as versões do aplicativo disponíveis com EMR Serverless 6.6.0.

Aplicativo	Version (Versão)
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMRNotas de versão inicial sem servidor

- EMRO Serverless é compatível com a classificação de configuração do Spark. `spark-defaults` Essa classificação altera os valores no `spark-defaults.conf` XML arquivo do Spark. As classificações de configuração permitem que você personalize aplicações. Para obter mais informações, consulte Configure applications.
- EMRO Serverless oferece suporte às classificações de configuração do Hive`hive-site`,, e. `tez-site` `emrfs-site` `core-site` Essa classificação pode alterar os valores no arquivo do

Hive, no `hive-site.xml` arquivo do Tez, EMRFS nas configurações EMR da Amazon ou no `tez-site.xml` arquivo do Hadoop, respectivamente. `core-site.xml` As classificações de configuração permitem que você personalize aplicações. Para obter mais informações, consulte Configure applications.

Alterações, aprimoramentos e problemas resolvidos específicos do motor

- A tabela a seguir lista os backports do Hive e do Tez.

  Mudanças no Hive e no Tez

Alteração	Descrição
Backport	TEZ-4430: Problema corrigido com a propriedade `tez.task.launch.cmd-opts`
Backport	HIVE-25971: Foram corrigidos os atrasos no desligamento da tarefa do Tez devido ao pool de threads em cache aberto

Histórico do documento

A tabela a seguir descreve as mudanças importantes na documentação desde a última versão do EMR Serverless. Para obter mais informações sobre as atualizações dessa documentação, você pode assinar um RSS feed.

Alteração	Descrição	Data
Nova versão	EMR Serverless 7.2.0	25 de julho de 2024
Nova versão	EMR Serverless 7.1.0	17 de abril de 2024
Atualização de uma política existente.	Adicionou o novo `Sid` `CloudWatchPolicyStatement` e `EC2PolicyStatement` à mazonEMRServerlessServiceRolePolicy política A.	25 de janeiro de 2024
Nova versão	EMR Serverless 7.0.0	29 de dezembro de 2023
Nova versão	EMR Serverless 6.15.0	17 de novembro de 2023
Novo atributo	Configure várias IAM funções a serem assumidas ao acessar buckets do Amazon S3 em contas diferentes do EMR Serverless (6.11 e superior)	18 de outubro de 2023
Nova versão	EMR Serverless 6.14.0	17 de outubro de 2023
Novo atributo	Configuração padrão do aplicativo para EMR Serverless	25 de setembro de 2023
Atualização para as propriedades padrão do Hive	Atualizados os valores padrão para `hive.driver.disk` , `hive.tez.disk.size`	12 de setembro de 2023

	`hive.tez.auto.reducer.parallelism`, e propriedades de trabalho do `tez.grouping.min-size` Hive.	
Nova versão	EMR Serverless 6.13.0	11 de setembro de 2023
Nova versão	EMR Serverless 6.12.0	21 de julho de 2023
Nova versão	EMR Serverless 6.11.0	8 de junho de 2023
Atualizar política de função vinculada ao serviço	Atualizou a `AmazonEMRServerlessServiceRolePolicy` SLRfunção para publicar o uso em nível de conta no `"AWS/Usage"` namespace.	20 de abril de 2023
EMR Serverless disponibilidade geral (GA)	Este é o primeiro lançamento público do EMR Serverless.	1º de junho de 2022

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.