Guía del usuario de Amazon EMR Serverless

Amazon EMR

Amazon EMR: Guía del usuario de Amazon EMR Serverless

Table of Contents

¿Qué es Amazon EMR Serverless?

Amazon EMR Serverless es una opción de implementación para Amazon EMR que proporciona un entorno de ejecución sin servidor. Esto simplifica el funcionamiento de las aplicaciones de análisis que utilizan los marcos de código abierto más recientes, como Apache Spark y Apache Hive. Con EMR Serverless, no es necesario configurar, optimizar, proteger ni operar clústeres para ejecutar aplicaciones con estos marcos.

EMRServerless le ayuda a evitar el aprovisionamiento excesivo o insuficiente de recursos para sus trabajos de procesamiento de datos. EMRServerless determina automáticamente los recursos que necesita la aplicación, los obtiene para procesar sus trabajos y los libera cuando estos finalizan. Para los casos de uso en los que las aplicaciones necesitan una respuesta en cuestión de segundos, como el análisis de datos interactivo, puede inicializar previamente los recursos que la aplicación necesita al crearla.

Con EMR Serverless, seguirás disfrutando de las ventajas de AmazonEMR, como la compatibilidad con código abierto, la simultaneidad y el rendimiento optimizado del tiempo de ejecución para marcos populares.

EMRServerless es adecuado para los clientes que desean utilizar aplicaciones con facilidad mediante marcos de código abierto. Ofrece un inicio rápido de los trabajos, una gestión automática de la capacidad y controles de costes sencillos.

Conceptos

En esta sección, abordamos los términos y conceptos de EMR Serverless que aparecen en nuestra Guía del usuario de EMR Serverless.

Versión de lanzamiento

Una EMR versión de Amazon es un conjunto de aplicaciones de código abierto del ecosistema de big data. Cada versión incluye diferentes aplicaciones, componentes y funciones de macrodatos que puede seleccionar para que EMR Serverless las implemente y configure de forma que puedan ejecutar sus aplicaciones. Al crear una aplicación, debe especificar su versión de lanzamiento. Elija la versión de EMR lanzamiento de Amazon y la versión del marco de código abierto que desee utilizar en su aplicación. Para obtener más información sobre las versiones preliminares, consulteVersiones de lanzamiento de Amazon EMR Serverless.

Aplicación

Con EMR Serverless, puede crear una o más aplicaciones EMR sin servidor que utilicen marcos de análisis de código abierto. Para crear una aplicación, debe especificar los siguientes atributos:

- La versión EMR de lanzamiento de Amazon para la versión del marco de código abierto que quieres usar. Para determinar la versión de lanzamiento, consulteVersiones de lanzamiento de Amazon EMR Serverless.
- El tiempo de ejecución específico que desea que utilice su aplicación, como Apache Spark o Apache Hive.

Después de crear una aplicación, puede enviar trabajos de procesamiento de datos o solicitudes interactivas a la aplicación.

Cada aplicación EMR sin servidor se ejecuta en una Amazon Virtual Private Cloud (VPC) segura, estrictamente separada del resto de aplicaciones. Además, puede utilizar AWS Identity and Access Management (IAM) políticas para definir qué usuarios y roles pueden acceder a la aplicación. También puede especificar límites para controlar y realizar un seguimiento de los costos de uso incurridos por la aplicación.

Considere la posibilidad de crear varias aplicaciones cuando necesite hacer lo siguiente:

- Utilice diferentes marcos de código abierto
- Utilice diferentes versiones de marcos de código abierto para diferentes casos de uso
- Realice pruebas A/B al actualizar de una versión a otra
- Mantenga entornos lógicos separados para los escenarios de prueba y producción
- Proporcione entornos lógicos separados para los diferentes equipos con controles de costos y seguimiento del uso independientes
- Separe line-of-business las diferentes aplicaciones

EMRServerless es un servicio regional que simplifica la forma en que las cargas de trabajo se ejecutan en varias zonas de disponibilidad de una región. Para obtener más información sobre cómo usar las aplicaciones con EMR Serverless, consulte. Interactuar con una aplicación

Ejecución de trabajo

La ejecución de un trabajo es una solicitud que se envía a una aplicación EMR sin servidor y que la aplicación ejecuta de forma asíncrona y realiza un seguimiento hasta su finalización. Algunos ejemplos de trabajos incluyen una consulta de HiveQL que se envía a una aplicación de Apache Hive o un script de procesamiento de datos que se envía a PySpark una aplicación de Apache Spark. Al enviar un trabajo, debe especificar un rol de tiempo de ejecución, creado en él, al que el trabajo IAM utilice para acceder AWS recursos, como objetos de Amazon S3. Puede enviar varias solicitudes de ejecución de tareas a una aplicación y cada ejecución de tareas puede utilizar una función de ejecución diferente para acceder AWS recursos. Una aplicación EMR sin servidor comienza a ejecutar trabajos en cuanto los recibe y ejecuta varias solicitudes de trabajo simultáneamente. Para obtener más información sobre cómo EMR Serverless ejecuta los trabajos, consulte. Trabajos en ejecución

Procesos de trabajo

Una aplicación EMR sin servidor utiliza trabajadores internamente para ejecutar sus cargas de trabajo. Los tamaños predeterminados de estos trabajadores se basan en el tipo de aplicación y en la versión de EMR lanzamiento de Amazon. Al programar la ejecución de un trabajo, puede anular estos tamaños.

Al enviar un trabajo, EMR Serverless calcula los recursos que la aplicación necesita para el trabajo y programa a los trabajadores. EMRServerless divide sus cargas de trabajo en tareas, descarga imágenes, aprovisiona y configura a los trabajadores, y los retira del servicio cuando finaliza el trabajo. EMRServerless amplía o reduce el número de trabajadores automáticamente en función de la carga de trabajo y el paralelismo necesarios en cada etapa del trabajo. Este escalado automático elimina la necesidad de estimar la cantidad de trabajadores que la aplicación necesita para ejecutar sus cargas de trabajo.

Capacidad preinicializada

EMRServerless proporciona una función de capacidad preinicializada que permite a los trabajadores inicializados y preparados para responder en cuestión de segundos. Esta capacidad crea de manera efectiva un grupo cálido de trabajadores para una aplicación. Para configurar esta función para cada aplicación, defina el `initial-capacity` parámetro de una aplicación. Al configurar la capacidad preinicializada, los trabajos pueden iniciarse inmediatamente para que pueda implementar aplicaciones iterativas y trabajos urgentes. Para obtener más información sobre los trabajadores preinicializados, consulte. Configuración de una aplicación

EMREstudio

EMRStudio es la consola de usuario que puede usar para administrar sus aplicaciones EMR sin servidor. Si no hay ningún EMR Studio en tu cuenta cuando creaste tu primera aplicación EMR sin servidor, crearemos uno automáticamente para ti. Puedes acceder a EMR Studio desde la EMR consola de Amazon o puedes activar el acceso federado desde tu proveedor de identidad (IdP) IAM a través IAM de Identity Center. Al hacerlo, los usuarios pueden acceder a Studio y gestionar las aplicaciones EMR sin servidor sin acceso directo a la EMR consola de Amazon. Para obtener más información sobre cómo funcionan las aplicaciones EMR sin servidor con EMR Studio, consulte Interactuar con la aplicación desde la consola de EMR Studio y. Ejecución de trabajos desde la consola de EMR Studio

Requisitos previos para empezar a usar Serverless EMR

Temas

Regístrate para obtener una Cuenta de AWS

Si no tienes un Cuenta de AWS, complete los pasos siguientes para crear uno.

Para suscribirte a una Cuenta de AWS

1. Abrir https://portal.aws.amazon.com/billing/registro.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en una Cuenta de AWS, un Usuario raíz de la cuenta de AWSse crea. El usuario root tiene acceso a todos Servicios de AWS y los recursos de la cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. En cualquier momento, puede ver la actividad de su cuenta actual y administrarla accediendo a https://aws.amazon.com/y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de suscribirse a una Cuenta de AWS, asegure su Usuario raíz de la cuenta de AWS, habilitar AWS IAM Identity Center y cree un usuario administrativo para no utilizar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión en la AWS Management Consolecomo propietario de la cuenta seleccionando el usuario root e introduciendo su Cuenta de AWS dirección de correo electrónico. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con un usuario root, consulte Iniciar sesión como usuario root en AWS Sign-In Guía del usuario.
2. Activa la autenticación multifactorial (MFA) para tu usuario root.

   Para obtener instrucciones, consulte Habilitar un MFA dispositivo virtual para su Cuenta de AWS usuario root (consola) en la Guía IAM del usuario.

Creación de un usuario con acceso administrativo

1. Habilite IAM Identity Center.

   Para obtener instrucciones, consulte Habilitar AWS IAM Identity Center en la AWS IAM Identity Center Guía del usuario.
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre el uso de Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la AWS IAM Identity Center Guía del usuario.

Iniciar sesión como usuario con acceso de administrador

- Para iniciar sesión con su usuario de IAM Identity Center, utilice el inicio de sesión URL que se envió a su dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario de IAM Identity Center, consulte Iniciar sesión en AWS acceda al portal en el AWS Sign-In Guía del usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos con privilegios mínimos.

Para obtener instrucciones, consulte Crear un conjunto de permisos en el AWS IAM Identity Center Guía del usuario.

2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para obtener instrucciones, consulte Añadir grupos en AWS IAM Identity Center Guía del usuario.

Concesión de permisos

En entornos de producción, le recomendamos que utilice políticas más específicas. Para ver ejemplos de dichas políticas, consulteEjemplos de políticas de acceso de usuarios para Serverless EMR. Para obtener más información sobre la administración del acceso, consulte Administración del acceso para AWS recursos en la Guía IAM del usuario.

Para los usuarios que necesiten empezar a usar EMR Serverless en un entorno sandbox, usen una política similar a la siguiente:

```
{
    "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 dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Crear un conjunto de permisos en el AWS IAM Identity Center Guía del usuario.

- Usuarios gestionados IAM a través de un proveedor de identidad:

  Cree un rol para la federación de identidades. Siga las instrucciones de la Guía del IAM usuario sobre cómo crear un rol para un proveedor de identidades externo (federación).

- IAMusuarios:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones de la Guía del IAMusuario sobre cómo crear un rol para un IAM usuario.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones de Añadir permisos a un usuario (consola) de la Guía del IAM usuario.

Concesión de acceso programático

Los usuarios necesitan acceso mediante programación si quieren interactuar con AWS fuera del AWS Management Console. La forma de conceder el acceso programático depende del tipo de usuario que acceda AWS.

Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Los usuarios se administran en IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI, AWS SDKs, o AWS APIs.	Siga las instrucciones de la interfaz que desea utilizar: • Para el registro AWS CLI, consulte Configuración del AWS CLI para usar AWS IAM Identity Center en la AWS Command Line Interface Guía del usuario. • En AWS SDKs, herramientas y AWS APIs, consulte la autenticación de IAM Identity Center en la AWS SDKsy la Guía de referencia de herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI, AWS SDKs, o AWS APIs.	Siguiendo las instrucciones de Uso de credenciales temporales con AWS los recursos de la Guía IAM del usuario.
IAM	(No recomendado) Utilice credenciales de larga duración para firmar las solicitudes programáticas	Siga las instrucciones de la interfaz que desea utilizar: • Para el registro AWS CLI, consulte Autenticación

¿Qué usuario necesita acceso programático?	Para	Mediante
	dirigidas al AWS CLI, AWS SDKs, o AWS APIs.	mediante credenciales IAM de usuario en AWS Command Line Interface Guía del usuario. • En AWS SDKsy herramientas, consulte Autenticarse con credenciales de larga duración en la AWS SDKsy la Guía de referencia de herramientas. • En AWS APIs, consulte Administrar las claves de acceso para IAM los usuarios en la Guía del IAM usuario.

Instale y configure el AWS CLI

Si desea utilizar EMR ServerlessAPIs, debe instalar la última versión del AWS Command Line Interface (AWS CLI). No necesitas el AWS CLI para usar EMR Serverless desde la consola de EMR Studio, y puedes empezar sin él CLI siguiendo los pasos que se indican enCómo empezar a usar EMR Serverless desde la consola.

Para configurar el AWS CLI

1. Para instalar la versión más reciente de AWS CLI para macOS, Linux o Windows, consulte Instalación o actualización de la última versión del AWS CLI.
2. Para configurar el AWS CLI y una configuración segura de su acceso a Servicios de AWS, incluido EMR Serverless, consulte Configuración rápida con `aws configure`.
3. Para verificar la configuración, introduzca el siguiente DataBrew comando en la línea de comandos.

```
aws emr-serverless help
```

AWS CLI los comandos utilizan el valor predeterminado Región de AWS de su configuración, a menos que la establezca con un parámetro o un perfil. Para configurar su Región de AWS con un parámetro, puede añadir el `--region` parámetro a cada comando.

Para configurar su Región de AWS con un perfil, añada primero un perfil con nombre en el `~/.aws/config` archivo o en el `%UserProfile%/.aws/config` archivo (para Microsoft Windows). Siga los pasos que se indican en los perfiles con nombre para AWS CLI. A continuación, configura tu Región de AWS y otros ajustes con un comando similar al del siguiente ejemplo.

```
[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 la consola de

La mayoría de los temas orientados a las consolas de esta sección comienzan en la consola de Amazon EMR. Si aún no has iniciado sesión en tu Cuenta de AWS, inicia sesión, abre la EMRconsola de Amazon y pasa a la siguiente sección para empezar a usar AmazonEMR.

Introducción a Amazon EMR Serverless

Este tutorial te ayuda a empezar a usar EMR Serverless al implementar un ejemplo de carga de trabajo de Spark o Hive. Crearás, ejecutarás y depurarás tu propia aplicación. Mostramos las opciones predeterminadas en la mayoría de las partes de este tutorial.

Antes de lanzar una aplicación EMR sin servidor, complete las siguientes tareas.

Temas

Otorgue permisos para usar Serverless EMR

Para usar EMR Serverless, necesitas un usuario o IAM rol con una política adjunta que otorgue permisos para EMR Serverless. Para crear un usuario y adjuntar la política adecuada a ese usuario, siga las instrucciones que se indican en. Concesión de permisos

Prepare el almacenamiento para EMR Serverless

En este tutorial, utilizarás un bucket de S3 para almacenar los archivos de salida y los registros de la carga de trabajo de ejemplo de Spark o Hive que ejecutarás con una aplicación sin EMR servidor. Para crear un depósito, sigue las instrucciones de la Guía del usuario de la consola Amazon Simple Storage Service Console. Sustituya cualquier referencia adicional *DOC-EXAMPLE-BUCKET* a por el nombre del depósito recién creado.

Cree un EMR estudio para ejecutar cargas de trabajo interactivas

Si quieres usar EMR Serverless para ejecutar consultas interactivas a través de libretas alojadas en EMR Studio, debes especificar un bucket de S3 y la función de servicio mínima de EMR Serverless

para crear un espacio de trabajo. Para ver los pasos de configuración, consulta Configurar un EMR estudio en la Guía de EMR administración de Amazon. Para obtener más información sobre las cargas de trabajo interactivas, consulteEjecute cargas de trabajo interactivas con EMR Serverless a través de Studio EMR.

Cree un rol de ejecución de tareas

Las ejecuciones de trabajos en EMR Serverless utilizan un rol de tiempo de ejecución que proporciona permisos granulares para aplicaciones específicas Servicios de AWS y recursos en tiempo de ejecución. En este tutorial, un depósito público de S3 aloja los datos y los scripts. El depósito *DOC-EXAMPLE-BUCKET* almacena la salida.

Para configurar un rol de tiempo de ejecución de un trabajo, primero cree un rol de tiempo de ejecución con una política de confianza para que EMR Serverless pueda usar el nuevo rol. A continuación, adjunte la política de acceso de S3 requerida a ese rol. Los siguientes pasos le guiarán a lo largo del proceso.

Console

1. Vaya a la consola IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, seleccione Roles.
3. Elija Crear rol.
4. Para el tipo de función, elija Política de confianza personalizada y pegue la siguiente política de confianza. Esto permite que los trabajos enviados a sus aplicaciones de Amazon EMR Serverless accedan a otros Servicios de AWS en su nombre.

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

5. Selecciona Siguiente para ir a la página Añadir permisos y, a continuación, selecciona Crear política.
6. La página Crear política se abre en una pestaña nueva. Pegue la política JSON a continuación.

> **Important**
>
> Sustituya *DOC-EXAMPLE-BUCKET* la política que aparece a continuación por el nombre real del bucket creado enPrepare el almacenamiento para EMR Serverless. Se trata de una política básica para el acceso a S3. Para ver más ejemplos de roles de ejecución de tareas, consulteFunciones de tiempo de ejecución de trabajos 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. En la página de revisión de la política, introduzca un nombre para la política, como`EMRServerlessS3AndGlueAccessPolicy`.
8. Actualice la página Adjuntar la política de permisos y elija`EMRServerlessS3AndGlueAccessPolicy`.
9. En la página Nombre, revisión y creación, en Nombre del rol, introduzca un nombre para el rol, por ejemplo,`EMRServerlessS3RuntimeRole`. Para crear este IAM rol, elija Crear rol.

CLI

1. Cree un archivo con un nombre e`mr-serverless-trust-policy.json` que contenga la política de confianza que se utilizará para el IAM rol. El archivo debe contener la siguiente política.

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

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

2. Cree un IAM rol con el nombre`EMRServerlessS3RuntimeRole`. Utilice la política de confianza que creó en el paso anterior.

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

En los resultados, anote el ARN. Utiliza el nuevo rol durante el envío ARN del trabajo, denominado después de esto el*job-role-arn*.

3. Cree un archivo con un nombre `emr-sample-access-policy.json` que defina la IAM política para su carga de trabajo. Esto proporciona acceso de lectura al script y a los datos almacenados en los depósitos públicos de S3 y acceso de lectura y escritura a. *DOC-EXAMPLE-BUCKET*

> **Important**
>
> Sustituya *DOC-EXAMPLE-BUCKET* la política que aparece a continuación por el nombre real del bucket creado en.. Prepare el almacenamiento para EMR Serverless Se trata de una política básica para AWS Acceso a Glue y S3. Para ver más ejemplos de roles de ejecución de tareas, consulteFunciones de tiempo de ejecución de trabajos 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. Cree una IAM política `EMRServerlessS3AndGlueAccessPolicy` con el nombre del archivo de política que creó en el paso 3. Anote lo que aparece ARN en el resultado, ya que utilizará ARN la nueva política en el siguiente paso.

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

Tenga en cuenta que la nueva política ARN aparece en el resultado. La sustituirás por ella *`policy-arn`* en el siguiente paso.

5. Adjunta la IAM política `EMRServerlessS3AndGlueAccessPolicy` al rol de ejecución del trabajo`EMRServerlessS3RuntimeRole`.

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

Cómo empezar a usar EMR Serverless desde la consola

Pasos que completar

- Paso 1: Crear una aplicación EMR sin servidor
- Paso 2: Envíe un trabajo, una ejecución o una carga de trabajo interactiva
- Paso 3: Ver la interfaz de usuario y los registros de la aplicación
- Paso 4: Limpiar

Paso 1: Crear una aplicación EMR sin servidor

Cree una nueva aplicación con EMR Serverless de la siguiente manera.

1. Inicie sesión en AWS Management Console y abre la EMR consola de Amazon en https://console.aws.amazon.com/emr.
2. En el panel de navegación izquierdo, selecciona EMRServerless para ir a la página de inicio de EMR Serverless.

3. Para crear o administrar aplicaciones EMR sin servidor, necesitas la interfaz de usuario de EMR Studio.

   - Si ya tienes un EMR estudio en Región de AWS donde quieras crear una aplicación, selecciona Administrar aplicaciones para ir a tu EMR estudio o selecciona el estudio que quieras usar.
   - Si no tienes un EMR estudio en Región de AWS donde desee crear una aplicación, elija Comenzar y, a continuación, elija Crear e inicie Studio. EMRServerless crea un EMR estudio para usted para que pueda crear y administrar aplicaciones.

4. En la interfaz de usuario de Create Studio que se abre en una nueva pestaña, introduce el nombre, el tipo y la versión de lanzamiento de la aplicación. Si solo desea ejecutar trabajos por lotes, seleccione Usar la configuración predeterminada solo para trabajos por lotes. Para las cargas de trabajo interactivas, selecciona Usar la configuración predeterminada para las cargas de trabajo interactivas. También puede ejecutar trabajos por lotes en aplicaciones con capacidad interactiva con esta opción. Si lo necesita, puede cambiar esta configuración más adelante.

   Para obtener más información, consulte Crear un estudio.

5. Seleccione Crear aplicación para crear su primera aplicación.

Continúe con la siguiente sección Paso 2: Envíe un trabajo, una ejecución o una carga de trabajo interactiva para enviar un trabajo, una ejecución o una carga de trabajo interactiva.

Paso 2: Envíe un trabajo, una ejecución o una carga de trabajo interactiva

Spark job run

En este tutorial, utilizamos un PySpark script para calcular el número de veces que aparecen palabras únicas en varios archivos de texto. Un bucket de S3 público y de solo lectura almacena tanto el script como el conjunto de datos.

Para ejecutar un trabajo de Spark

1. Carga el script de muestra `wordcount.py` en tu nuevo depósito con el siguiente comando.

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

2. Si lo Paso 1: Crear una aplicación EMR sin servidor completa, accederá a la página de detalles de la aplicación en EMR Studio. Allí, selecciona la opción Enviar trabajo.
3. En la página Enviar trabajo, complete lo siguiente.
   - En el campo Nombre, introduzca el nombre con el que quiere llamar a la ejecución del trabajo.
   - En el campo Función en tiempo de ejecución, introduzca el nombre de la función en la que la creóCree un rol de ejecución de tareas.
   - En el campo Ubicación del script, `s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py` introdúzcalo como S3URI.
   - En el campo Argumentos del script, introduzca`["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/output"]`.
   - En la sección de propiedades de Spark, selecciona Editar como texto e introduce las siguientes configuraciones.

     ```
     --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 la ejecución del trabajo, selecciona Enviar trabajo.
5. En la pestaña Trabajos ejecutados, debería ver el nuevo trabajo ejecutándose con el estado En ejecución.

Hive job run

En esta parte del tutorial, creamos una tabla, insertamos algunos registros y ejecutamos una consulta de agregación de recuentos. Para ejecutar el trabajo de Hive, primero cree un archivo que contenga todas las consultas de Hive para ejecutarlas como parte de un solo trabajo, cargue el archivo en S3 y especifique esta ruta de S3 al iniciar el trabajo de Hive.

Para ejecutar un trabajo de Hive

1. Cree un archivo llamado `hive-query.ql` que contenga todas las consultas que desee ejecutar en su trabajo de Hive.

   ```
   create database if not exists emrserverless;
   use emrserverless;
   create table if not exists test_table(id int);
   ```

```
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. `hive-query.ql`Cárguelo en su bucket de S3 con el siguiente comando.

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

3. Si lo Paso 1: Crear una aplicación EMR sin servidor completa, accederá a la página de detalles de la aplicación en EMR Studio. Allí, selecciona la opción Enviar trabajo.
4. En la página Enviar trabajo, complete lo siguiente.
   - En el campo Nombre, introduzca el nombre con el que quiere llamar a la ejecución del trabajo.
   - En el campo Función en tiempo de ejecución, introduzca el nombre de la función en la que la creóCree un rol de ejecución de tareas.
   - En el campo Ubicación del script, `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql` introdúzcalo como S3URI.
   - En la sección de propiedades de la colmena, elija Editar como texto e introduzca las siguientes configuraciones.

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

   - En la sección Configuración del trabajo, elija Editar como JSON e introduzca lo siguienteJSON.

```
{
   "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 la ejecución del trabajo, seleccione Enviar trabajo.
6. En la pestaña Trabajos ejecutados, debería ver el nuevo trabajo ejecutándose con el estado En ejecución.

Interactive workload

Con Amazon EMR 6.14.0 y versiones posteriores, puedes usar libretas alojadas en EMR Studio para ejecutar cargas de trabajo interactivas para Spark en Serverless. EMR Para obtener más información, incluidos los permisos y requisitos previos, consulte. Ejecute cargas de trabajo interactivas con EMR Serverless a través de Studio EMR

Una vez que haya creado la aplicación y configurado los permisos necesarios, siga los siguientes pasos para ejecutar un bloc de notas interactivo con EMR Studio:

1. Ve a la pestaña Espacios de trabajo en EMR Studio. Si aún necesita configurar una ubicación de almacenamiento de Amazon S3 y un rol de servicio de EMR Studio, seleccione el botón Configurar estudio en el encabezado de la parte superior de la pantalla.
2. Para acceder a un bloc de notas, selecciona un espacio de trabajo o crea uno nuevo. Usa el inicio rápido para abrir tu espacio de trabajo en una pestaña nueva.
3. Ve a la pestaña recién abierta. Selecciona el icono de cómputo en la barra de navegación de la izquierda. Seleccione EMR Serverless como tipo de procesamiento.
4. Seleccione la aplicación con capacidad interactiva que creó en la sección anterior.
5. En el campo Función de ejecución, introduzca el nombre de la IAM función que la aplicación EMR sin servidor puede asumir durante la ejecución de la tarea. Para obtener más información sobre las funciones en tiempo de ejecución, consulte Funciones en tiempo de ejecución de Job en la Guía del usuario de Amazon EMR Serverless.
6. Seleccione Adjuntar. Esto puede tardar hasta un minuto. La página se actualizará cuando se adjunte.
7. Escoja un núcleo e inicie un cuaderno. También puedes buscar ejemplos de cuadernos en EMR Serverless y copiarlos en tu espacio de trabajo. Para acceder a las libretas de ejemplo, navega hasta el **`{...}`**menú de navegación de la izquierda y busca entre las libretas que tengan `serverless` el nombre del archivo de la libreta.

8. En el cuaderno, puedes acceder al enlace del registro de conductores y a un enlace a la interfaz de usuario de Apache Spark, una interfaz en tiempo real que proporciona métricas para supervisar tu trabajo. Para obtener más información, consulte Supervisión de aplicaciones y trabajos EMR sin servidor en la Guía del usuario de Amazon EMR Serverless.

Al adjuntar una aplicación a un espacio de trabajo de Studio, el inicio de la aplicación se activa automáticamente si aún no se está ejecutando. También puedes iniciar previamente la aplicación y tenerla lista antes de adjuntarla al espacio de trabajo.

Paso 3: Ver la interfaz de usuario y los registros de la aplicación

Para ver la interfaz de usuario de la aplicación, primero identifique la tarea ejecutada. Hay disponible una opción para la interfaz de usuario de Spark o la interfaz de usuario de Hive Tez en la primera fila de opciones para la ejecución de ese trabajo, según el tipo de trabajo. Selecciona la opción adecuada.

Si has elegido la interfaz de usuario de Spark, selecciona la pestaña Ejecutores para ver los registros de los controladores y ejecutores. Si has elegido la interfaz de usuario de Hive Tez, selecciona la pestaña Todas las tareas para ver los registros.

Una vez que el estado de ejecución de la tarea se muestre como correcta, podrá ver el resultado de la tarea en su bucket de S3.

Paso 4: Limpiar

Si bien la aplicación que creó debería detenerse automáticamente después de 15 minutos de inactividad, le recomendamos que libere los recursos que no tenga intención de volver a utilizar.

Para eliminar la aplicación, vaya a la página Listar aplicaciones. Seleccione la aplicación que ha creado y elija Acciones → Detener para detenerla. Cuando la aplicación esté en ese `STOPPED` estado, seleccione la misma aplicación y elija Acciones → Eliminar.

Para ver más ejemplos de cómo ejecutar tareas de Spark y Hive, consulte Empleos en Spark yEmpleos en Hive.

Cómo empezar desde AWS CLI

Paso 1: Crear una aplicación EMR sin servidor

Utilice el `emr-serverless create-application` comando para crear su primera aplicación EMR sin servidor. Debes especificar el tipo de aplicación y la etiqueta de EMR lanzamiento de Amazon asociada a la versión de la aplicación que quieres usar. El nombre de la aplicación es opcional.

Spark

Para crear una aplicación Spark, ejecuta el siguiente comando.

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

Hive

Para crear una aplicación Hive, ejecuta el siguiente comando.

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

Anote el ID de la aplicación devuelto en el resultado. Utilizará el ID para iniciar la solicitud y durante la presentación del puesto de trabajo, que en adelante se denominará*`application-id`*.

Antes de continuarPaso 2: envíe una ejecución de trabajo a su aplicación EMR sin servidor, asegúrese de que su solicitud haya llegado al `CREATED` estado con el `get-application`API.

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

EMRServerless crea trabajadores para adaptarse a los trabajos solicitados. De forma predeterminada, se crean bajo demanda, pero también puede especificar una capacidad

preinicializada configurando el `initialCapacity` parámetro al crear la aplicación. También puede limitar la capacidad máxima total que puede utilizar una aplicación con el `maximumCapacity` parámetro. Para más información sobre estas opciones, consulte Configuración de una aplicación.

Paso 2: envíe una ejecución de trabajo a su aplicación EMR sin servidor

Ahora su aplicación EMR sin servidor está lista para ejecutar tareas.

Spark

En este paso, utilizamos un PySpark script para calcular el número de veces que aparecen palabras únicas en varios archivos de texto. Un bucket de S3 público y de solo lectura almacena tanto el script como el conjunto de datos. La aplicación envía el archivo de salida y los datos de registro del entorno de ejecución de Spark a `/output` los `/logs` directorios del depósito de S3 que creaste.

Para ejecutar un trabajo de Spark

1. Usa el siguiente comando para copiar el script de muestra que ejecutaremos en tu nuevo bucket.

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

2. En el siguiente comando, *`application-id`* sustitúyalo por el ID de la aplicación. *`job-role-arn`*Sustitúyalo por el rol de tiempo de ejecución en el ARN que lo creasteCree un rol de ejecución de tareas. Sustituya *`job-run-name`* con el nombre que quieras llamar a tu trabajo. Sustituya todas *`DOC-EXAMPLE-BUCKET`* las cadenas por el bucket de Amazon S3 que creó y añádalas `/output` a la ruta. Esto crea una nueva carpeta en su bucket donde EMR Serverless puede copiar los archivos de salida de su aplicación.

```
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. Anote el ID de ejecución del trabajo devuelto en el resultado. *job-run-id*Sustitúyalo por este ID en los pasos siguientes.

Hive

En este tutorial, creamos una tabla, insertamos algunos registros y ejecutamos una consulta de agregación de recuentos. Para ejecutar el trabajo de Hive, primero cree un archivo que contenga todas las consultas de Hive para ejecutarlas como parte de un solo trabajo, cargue el archivo en S3 y especifique esta ruta de S3 al iniciar el trabajo de Hive.

Para ejecutar un trabajo de Hive

1. Cree un archivo llamado `hive-query.ql` que contenga todas las consultas que desee ejecutar en su trabajo de Hive.

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

2. `hive-query.ql`Cárguelo en su bucket de S3 con el siguiente comando.

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

3. En el siguiente comando, *application-id* sustitúyalo por su propio ID de aplicación. *job-role-arn*Sustitúyalo por el rol de tiempo de ejecución en el ARN que lo creasteCree un rol de ejecución de tareas. Sustituya todas *DOC-EXAMPLE-BUCKET* las cadenas por el bucket de Amazon S3 que creó y añada `/output` and `/logs` a la ruta. Esto crea nuevas carpetas en su bucket, donde EMR Serverless puede copiar los archivos de salida y registro de su aplicación.

```
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. Anote el ID de ejecución del trabajo devuelto en el resultado. *job-run-id*Sustitúyalo por este ID en los pasos siguientes.

Paso 3: Revise el resultado de la ejecución de la tarea

La ejecución del trabajo suele tardar entre 3 y 5 minutos en completarse.

Spark

Puedes comprobar el estado de tu trabajo de Spark con el siguiente comando.

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

```
    --job-run-id job-run-id
```

Con el destino del registro establecido en `s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/logs`, puedes encontrar los registros de este trabajo específico en el que se está ejecutando `s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/logs/applications/application-id/jobs/job-run-id`.

En el caso de las aplicaciones de Spark, EMR Serverless envía los registros de eventos cada 30 segundos a la `sparklogs` carpeta de destino del registro de S3. Cuando termines tu trabajo, los registros de tiempo de ejecución de Spark correspondientes al controlador y los ejecutores se cargan en carpetas con el nombre adecuado según el tipo de trabajador, como o. `driver` `executor` El resultado del PySpark trabajo se carga en. `s3://DOC-EXAMPLE-BUCKET/output/`

Hive

Puede comprobar el estado de su trabajo de Hive con el siguiente comando.

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

Con el destino del registro establecido en `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs`, puede encontrar los registros de este trabajo específico en el que se está `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id` ejecutando.

En el caso de las aplicaciones de Hive, EMR Serverless carga continuamente el controlador de Hive a la `HIVE_DRIVER` carpeta y Tez asigna los registros a la `TEZ_TASK` carpeta de destino del registro de S3. Cuando la ejecución del trabajo alcance el `SUCCEEDED` estado, el resultado de la consulta de Hive estará disponible en la ubicación de Amazon S3 que especificó en el `monitoringConfiguration` campo `deconfigurationOverrides`.

Paso 4: Limpiar

Cuando termines de trabajar con este tutorial, considera la posibilidad de eliminar los recursos que has creado. Le recomendamos que publique los recursos que no tenga intención de volver a utilizar.

Elimine su aplicación

Para eliminar una aplicación, utilice el siguiente comando.

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

Elimine su depósito de registro de S3

Para eliminar el depósito de registro y salida de S3, utilice el siguiente comando. *DOC-EXAMPLE-BUCKET*Sustitúyalo por el nombre real del bucket de S3 creado enPrepare el almacenamiento para EMR Serverless..

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

Elimine el rol de ejecución de su trabajo

Para eliminar el rol de tiempo de ejecución, separe la política del rol. A continuación, puede eliminar tanto el rol como la política.

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

Para eliminar el rol, usa el siguiente comando.

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

Para eliminar la política asociada a la función, utilice el siguiente comando.

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

Para ver más ejemplos de cómo ejecutar trabajos de Spark y Hive, consulta Empleos en Spark yEmpleos en Hive.

Interactuar con una aplicación

En esta sección, se explica cómo puede interactuar con su aplicación Amazon EMR Serverless con AWS CLI y los valores predeterminados para los motores Spark y Hive.

Temas

Estados de la aplicación

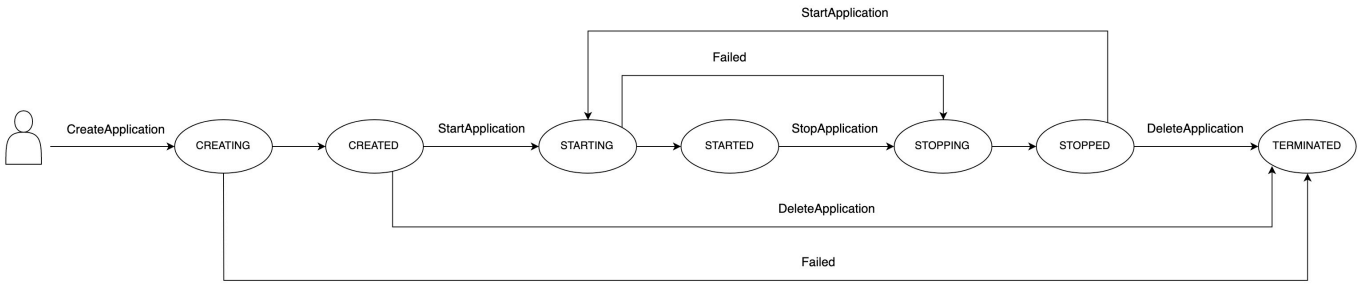
Al crear una aplicación con EMR Serverless, la ejecución de la aplicación pasa a ese estado. `CREATING` A continuación, pasa por los estados siguientes hasta que termina de ejecutarse correctamente (finaliza con el código `0`) o no (finaliza con un código distinto de cero).

Las aplicaciones pueden tener los siguientes estados:

Estado	Descripción
Creando	La aplicación se está preparando y aún no está lista para usarse.
Creado	La aplicación se creó, pero aún no ha aprovisio nado su capacidad. Puede modificar la aplicación para cambiar su configuración de capacidad inicial.
Iniciando	La aplicación se está iniciando y aprovisio nando capacidad.

Estado	Descripción
Empezado	La solicitud está lista para aceptar nuevos trabajos. La aplicación solo acepta trabajos cuando se encuentra en este estado.
Deteniendo	Se han completado todos los trabajos y la solicitud está agotando su capacidad.
Stopped	La aplicación está detenida y no hay recursos en ejecución en ella. Puede modificar la aplicación para cambiar su configuración de capacidad inicial.
Terminado	La solicitud se ha cancelado y no aparece en su lista de solicitudes.

El siguiente diagrama muestra la trayectoria de los estados de las aplicaciones EMR sin servidor.



Interactuar con la aplicación desde la consola de EMR Studio

Desde la consola de EMR Studio, puede crear, ver y gestionar aplicaciones EMR sin servidor. Para ir a la consola de EMR Studio, sigue las instrucciones de Cómo empezar desde la consola.

Crear una aplicación

En la página Crear aplicación, puede crear una aplicación EMR sin servidor siguiendo estos pasos.

1. En el campo Nombre, introduzca el nombre con el que desea llamar a la aplicación.

2. En el campo Tipo, selecciona Spark o Hive como tipo de aplicación.
3. En el campo Versión de lanzamiento, selecciona el número de EMR versión.
4. En las opciones de arquitectura, elija la arquitectura del conjunto de instrucciones que desee utilizar. Para obtener más información, consulte Opciones de arquitectura Amazon EMR Serverless.

   - arm64: ARM arquitectura de 64 bits; para usar procesadores Graviton
   - x86_64: arquitectura x86 de 64 bits; para utilizar procesadores basados en x86
5. Hay dos opciones de configuración de la aplicación para los campos restantes: configuración predeterminada y configuración personalizada. Estos campos son opcionales.

   Configuración predeterminada: la configuración predeterminada le permite crear una aplicación rápidamente con una capacidad preinicializada. Esto incluye un controlador y un ejecutor para Spark, y un controlador y un Tez Task para Hive. La configuración predeterminada no habilita la conectividad de red con su. VPCs La aplicación está configurada para detenerse si está inactiva durante 15 minutos y se inicia automáticamente al enviar el trabajo.

   Configuración personalizada: la configuración personalizada le permite modificar las siguientes propiedades.

   - Capacidad preinicializada: el número de conductores y ejecutores o trabajadores de Hive Tez Task y el tamaño de cada trabajador.
   - Límites de aplicación: la capacidad máxima de una aplicación.
   - Comportamiento de la aplicación: comportamiento de inicio y parada automáticos de la aplicación.
   - Conexiones de red: conectividad de red a los recursosVPC.
   - Etiquetas: etiquetas personalizadas que puede asignar a la aplicación.

   Para obtener más información sobre la capacidad preinicializada, los límites de las aplicaciones y el comportamiento de las aplicaciones, consulte. Configuración de una aplicación Para obtener más información sobre la conectividad de red, consulte. Configurar VPC el acceso
6. Para crear la aplicación, seleccione Crear aplicación.

Enumerar aplicaciones

Puede ver todas las aplicaciones EMR sin servidor existentes en la página Listar aplicaciones. Puede elegir el nombre de una aplicación para ir a la página de detalles de esa aplicación.

Administración de aplicaciones

Puede realizar las siguientes acciones en una aplicación desde la página Listar aplicaciones o desde la página de detalles de una aplicación específica.

Iniciar la aplicación

Seleccione esta opción para iniciar manualmente una aplicación.

Detenga la aplicación

Seleccione esta opción para detener manualmente una aplicación. Una aplicación no debe tener ningún trabajo en ejecución para poder detenerla. Para obtener más información sobre las transiciones de estado de las aplicaciones, consulteEstados de la aplicación.

Configurar la aplicación

Edite los ajustes opcionales de una aplicación desde la página Configurar la aplicación. Puede cambiar la mayoría de los ajustes de la aplicación. Por ejemplo, puede cambiar la etiqueta de lanzamiento de una aplicación para actualizarla a una versión diferente de AmazonEMR, o puede cambiar la arquitectura de x86_64 a arm64. Las demás configuraciones opcionales son las mismas que las que se encuentran en la sección Configuración personalizada de la página Crear aplicación. Para obtener más información sobre la configuración de la aplicación, consulteCrear una aplicación.

Eliminar la aplicación

Seleccione esta opción para eliminar manualmente una aplicación. Debe detener una aplicación para poder eliminarla. Para obtener más información sobre las transiciones de estado de las aplicaciones, consulteEstados de la aplicación.

Interactúa con la aplicación en el AWS CLI

Desde el AWS CLI, puede crear, describir y eliminar aplicaciones individuales. También puede enumerar todas sus aplicaciones para verlas de un vistazo. En esta sección se describe cómo realizar estas acciones. Para obtener más información sobre las operaciones de la aplicación, como

iniciar, detener y actualizar la aplicación, consulte la APIReferencia sobre EMR sistemas sin servidor. Para ver ejemplos de cómo usar EMR Serverless mediante API el AWS SDK for Java, consulte los ejemplos de Java en nuestro GitHub repositorio. Para ver ejemplos de cómo usar EMR Serverless API usando el AWS SDK for Python (Boto), consulta los ejemplos de Python en nuestro GitHub repositorio.

Para crear una aplicación, utilice`create-application`. Debe especificar `SPARK` o `HIVE` como aplicación`type`. Este comando devuelve el nombre y ARN el identificador de la aplicación.

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

Para describir una aplicación, utilice `get-application` y proporcione su`application-id`. Este comando devuelve las configuraciones relacionadas con el estado y la capacidad de la aplicación.

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

Para ver una lista de todas sus aplicaciones, llame`list-applications`. Este comando devuelve las mismas propiedades que todas las aplicaciones`get-application`, pero las incluye.

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

Para eliminar su solicitud, llame `delete-application` y proporcione su`application-id`.

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

Configuración de una aplicación

Con EMR Serverless, puede configurar las aplicaciones que utiliza. Por ejemplo, puede establecer la capacidad máxima a la que puede ampliarse una aplicación, configurar la capacidad preinicializada para que los conductores y los trabajadores estén preparados para responder y especificar un conjunto común de configuraciones de tiempo de ejecución y supervisión a nivel de aplicación. En las siguientes páginas se describe cómo configurar las aplicaciones cuando se utiliza EMR Serverless.

Temas

Comprender el comportamiento de las aplicaciones

Comportamiento predeterminado de la aplicación

Inicio automático: una aplicación está configurada de forma predeterminada para que se inicie automáticamente al enviar el trabajo. Puede desactivar esta función.

Parada automática: una aplicación está configurada de forma predeterminada para que se detenga automáticamente cuando está inactiva durante 15 minutos. Cuando una aplicación cambia al `STOPPED` estado, libera cualquier capacidad preinicializada configurada. Puede modificar la cantidad de tiempo de inactividad antes de que una aplicación se detenga automáticamente o puede desactivar esta función.

Capacidad máxima

Puede configurar la capacidad máxima a la que puede ampliarse una aplicación. Puede especificar la capacidad máxima en términos de CPU memoria (GB) y disco (GB).

> **Note**
>
> Se recomienda configurar la capacidad máxima para que sea proporcional al tamaño de los trabajadores admitidos multiplicando el número de trabajadores por su tamaño. Por ejemplo, si desea limitar la aplicación a 50 trabajadores con 2vCPUs, 16 GB de memoria y 20 GB de disco, establezca la capacidad máxima en 100vCPUs, 800 GB de memoria y 1000 GB de disco.

Configuraciones de trabajo compatibles

En la siguiente tabla se muestran las configuraciones y los tamaños de trabajadores compatibles que puede especificar para EMR Serverless. Puede configurar diferentes tamaños para los controladores y los ejecutores en función de las necesidades de su carga de trabajo.

CPU	Memoria	Almacenamiento efímero predeterminado
1 v CPU	Mínimo 2 GB, máximo 8 GB, en incrementos de 1 GB	20 GB - 200 GB
2 v CPU	Mínimo 4 GB, máximo 16 GB, en incrementos de 1 GB	20 GB - 200 GB
4 v CPU	Mínimo 8 GB, máximo 30 GB, en incrementos de 1 GB	20 GB - 200 GB
8 v CPU	Mínimo 16 GB, máximo 60 GB, en incrementos de 4 GB	20 GB - 200 GB
16 v CPU	Mínimo 32 GB, máximo 120 GB, en incrementos de 8 GB	20 GB - 200 GB

CPU— Cada trabajador puede tener 1, 2, 4, 8 o 16vCPUs.

Memoria: cada trabajador tiene memoria, especificada en GB, dentro de los límites indicados en la tabla anterior. Los trabajos de Spark tienen una sobrecarga de memoria, lo que significa que la memoria que utilizan es superior a los tamaños de contenedor especificados. Esta sobrecarga se especifica con las propiedades `spark.driver.memory0verhead` y`spark.executor.memory0verhead`. La sobrecarga tiene un valor predeterminado del 10% de la memoria del contenedor, con un mínimo de 384 MB. Debe tener en cuenta esta sobrecarga al elegir el tamaño de los trabajadores.

Por ejemplo, si eliges 4 vCPUs para tu instancia de trabajo y una capacidad de almacenamiento preinicializada de 30 GB, debes establecer un valor de aproximadamente 27 GB como memoria ejecutora para tu trabajo de Spark. Esto maximiza el uso de la capacidad preinicializada. La memoria utilizable sería de 27 GB, más un 10% de 27 GB (2,7 GB), para un total de 29,7 GB.

Disco: puede configurar a cada trabajador con discos de almacenamiento temporal con un tamaño mínimo de 20 GB y máximo de 200 GB. Solo paga por el almacenamiento adicional de más de 20 GB que configure por trabajador.

Capacidad preinicializada

EMRLa tecnología Serverless ofrece una función opcional que permite que el conductor y los trabajadores estén preinicializados y listos para responder en cuestión de segundos. Esto crea, de manera efectiva, un grupo cálido de trabajadores para una aplicación. Esta función se denomina capacidad preinicializada. Para configurar esta función, puede establecer el `initialCapacity` parámetro de una aplicación en función del número de trabajadores que desee preinicializar. Con la capacidad de trabajo preinicializada, los trabajos comienzan inmediatamente. Esto resulta ideal cuando se desean implementar aplicaciones iterativas y trabajos urgentes.

Al enviar un trabajo, si `initialCapacity` hay trabajadores disponibles, el trabajo utiliza esos recursos para iniciar su ejecución. Si esos trabajadores ya están siendo utilizados por otros trabajos, o si el trabajo necesita más recursos de los disponibles`initialCapacity`, la solicitud solicita y obtiene trabajadores adicionales, hasta el límite máximo de recursos establecido para la solicitud. Cuando un trabajo termina de ejecutarse, libera a los trabajadores que utilizó y el número de recursos disponibles para la aplicación vuelve a ser igual`initialCapacity`. Una aplicación mantiene los recursos incluso después `initialCapacity` de que los trabajos terminen de ejecutarse. La aplicación libera los recursos sobrantes `initialCapacity` cuando los trabajos ya no los necesitan para ejecutarse.

La capacidad preinicializada está disponible y lista para usarse cuando se inicie la aplicación. La capacidad preinicializada se vuelve inactiva cuando se detiene la aplicación. Una aplicación pasa a ese `STARTED` estado solo si la capacidad preinicializada solicitada se ha creado y está lista para usarse. Durante todo el tiempo que la aplicación esté en ese `STARTED` estado, EMR Serverless mantiene la capacidad preinicializada disponible para ser utilizada o utilizada por tareas o cargas de trabajo interactivas. La función restaura la capacidad de los contenedores liberados o averiados. Esto mantiene el número de trabajadores que especifica el `InitialCapacity` parámetro. El estado de una aplicación sin capacidad preinicializada puede cambiar inmediatamente de a`CREATED`. `STARTED`

Puede configurar la aplicación para que libere capacidad preinicializada si no se utiliza durante un período de tiempo determinado, con un valor predeterminado de 15 minutos. Una solicitud detenida se inicia automáticamente cuando envías un nuevo trabajo. Puede establecer estas configuraciones de inicio y parada automáticas al crear la solicitud, o puede cambiarlas cuando la aplicación esté en un `STOPPED` estado `CREATED` o.

Puede cambiar los `InitialCapacity` recuentos y especificar configuraciones informáticasCPU, como la memoria y el disco, para cada trabajador. Como no puede realizar modificaciones parciales,

debe especificar todas las configuraciones informáticas al cambiar los valores. Solo puede cambiar las configuraciones cuando la aplicación esté en el `STOPPED` estado `CREATED` o.

> **Note**
>
> Para optimizar el uso de los recursos de su aplicación, le recomendamos alinear los tamaños de los contenedores con los tamaños de los trabajadores con capacidad preinicializados. Por ejemplo, si configuras el tamaño de tu ejecutor de Spark en 2 CPUs y tu memoria en 8 GB, pero el tamaño del trabajador con capacidad preinicializado es de 4 CPUs con 16 GB de memoria, los ejecutores de Spark solo utilizan la mitad de los recursos de los trabajadores cuando se les asigna este trabajo.

Personalización de la capacidad preinicializada de Spark y Hive

Puede personalizar aún más la capacidad preinicializada para las cargas de trabajo que se ejecutan en marcos de big data específicos. Por ejemplo, cuando una carga de trabajo se ejecuta en Apache Spark, puede especificar cuántos trabajadores comienzan como conductores y cuántos comienzan como ejecutores. Del mismo modo, cuando usas Apache Hive, puedes especificar cuántos trabajadores comienzan como conductores de Hive y cuántos deben ejecutar las tareas de Tez.

Configurar una aplicación que ejecute Apache Hive con capacidad preinicializada

La siguiente API solicitud crea una aplicación que ejecuta Apache Hive basada en la EMR versión emr-6.6.0 de Amazon. La aplicación comienza con 5 controladores Hive preinicializados, cada uno con 2 v CPU y 4 GB de memoria, y 50 workworkers Tez preinicializados, cada uno con 4 v y 8 GB de memoria. CPU Cuando las consultas de Hive se ejecutan en esta aplicación, primero utilizan los elementos de trabajo preinicializados y comienzan a ejecutarse inmediatamente. Si todos los trabajadores preinicializados están ocupados y se envían más trabajos de Hive, la aplicación puede ampliarse hasta un total de 400 v CPU y 1024 GB de memoria. Si lo desea, puede omitir la capacidad del trabajador o del `DRIVER` trabajador. `TEZ_TASK`

```
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"
}'
```

Configurar una aplicación que ejecute Apache Spark con capacidad preinicializada

La siguiente API solicitud crea una aplicación que ejecuta Apache Spark 3.2.0 basada en la EMR versión 6.6.0 de Amazon. La aplicación comienza con 5 controladores Spark preinicializados, cada uno con 2 v CPU y 4 GB de memoria, y 50 ejecutores preinicializados, cada uno con 4 v y 8 GB de memoria. CPU Cuando los trabajos de Spark se ejecutan en esta aplicación, primero utilizan los elementos de trabajo preinicializados y comienzan a ejecutarse inmediatamente. Si todos los trabajadores preinicializados están ocupados y se envían más trabajos a Spark, la solicitud puede ampliarse hasta un total de 400 v CPU y 1024 GB de memoria. Si lo desea, puede omitir la capacidad del o del`DRIVER`. `EXECUTOR`

> **Note**
>
> Spark añade una sobrecarga de memoria configurable, con un valor predeterminado del 10%, a la memoria solicitada para el controlador y los ejecutores. Para que los trabajos utilicen elementos de trabajo preinicializados, la configuración de memoria de capacidad inicial debe ser mayor que la memoria que solicitan el trabajo y la 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"
  }'
```

Configuración de aplicaciones predeterminada para Serverless EMR

Puede especificar un conjunto común de configuraciones de tiempo de ejecución y supervisión a nivel de aplicación para todos los trabajos que envíe en la misma aplicación. Esto reduce la sobrecarga adicional asociada a la necesidad de enviar las mismas configuraciones para cada trabajo.

Puede modificar las configuraciones en los siguientes momentos:

- Declare las configuraciones a nivel de aplicación al enviar el trabajo.
- Anule las configuraciones predeterminadas durante la ejecución del trabajo.

En las siguientes secciones se proporcionan más detalles y un ejemplo para ampliar el contexto.

Declarar las configuraciones a nivel de aplicación

Puede especificar las propiedades de configuración de registro y tiempo de ejecución a nivel de aplicación para los trabajos que envíe en la aplicación.

`monitoringConfiguration`

Para especificar las configuraciones de registro de los trabajos que envíe con la aplicación, utilice el `monitoringConfiguration`campo. Para obtener más información sobre el registro de EMR Serverless, consulteAlmacenamiento de registros.

`runtimeConfiguration`

Para especificar propiedades de configuración en tiempo de ejecución, por ejemplo`spark-defaults`, proporcione un objeto de configuración en el `runtimeConfiguration` campo. Esto afecta a las configuraciones predeterminadas de todos los trabajos que envíe con la aplicación. Para obtener más información, consulte Parámetro de anulación de la configuración de Hive y Parámetro de anulación de la configuración de Spark.

Las clasificaciones de configuración disponibles varían según la versión específica de EMR Serverless. Por ejemplo, las clasificaciones para el Log4j personalizado solo `spark-executor-log4j2` están disponibles en las versiones 6.8.0 `spark-driver-log4j2` y superiores. Para obtener una lista de propiedades específicas de la aplicación, consulte y. Aumente las propiedades laborales Hive las propiedades de los trabajos

También puede configurar las propiedades de Apache Log4j2, AWS Secrets Manager para la protección de datos y el tiempo de ejecución de Java 17 a nivel de aplicación.

Para transmitir los secretos de Secrets Manager a nivel de aplicación, adjunte la siguiente política a los usuarios y roles que necesiten crear o actualizar aplicaciones EMR sin servidor con secretos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerPolicy",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:secretsmanager:your-secret-arn"
        }
    ]
}
```

Para obtener más información sobre la creación de políticas personalizadas para los secretos, consulte los ejemplos de políticas de permisos para AWS Secrets Manager en la AWS Secrets Manager Guía del usuario.

> **Note**
>
> Lo `runtimeConfiguration` que especifique a nivel de aplicación se asigna a `applicationConfiguration` `StartJobRun`API.

Ejemplo de declaración

El siguiente ejemplo muestra cómo declarar las configuraciones predeterminadas con`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
        }
    }'
```

Anular las configuraciones durante la ejecución de un trabajo

Puede especificar las anulaciones de configuración para la configuración de la aplicación y la configuración de supervisión con. `StartJobRun`API EMRA continuación, Serverless combina las configuraciones que especifique a nivel de aplicación y a nivel de trabajo para determinar las configuraciones para la ejecución del trabajo.

El nivel de granularidad cuando se produce la fusión es el siguiente:

- **`ApplicationConfiguration`**- Por ejemplo`spark-defaults`, el tipo de clasificación.
- **`MonitoringConfiguration`**- Por ejemplo, el tipo de configuración`s3MonitoringConfiguration`.

> **Note**
>
> La prioridad de las configuraciones que proporcione `StartJobRun`sustituirá a las configuraciones que proporcione a nivel de aplicación.

Para obtener más información sobre las clasificaciones de prioridades, consulte Parámetro de anulación de la configuración de Hive yParámetro de anulación de la configuración de Spark.

Al iniciar un trabajo, si no especifica una configuración determinada, se heredará de la aplicación. Si declara las configuraciones a nivel de trabajo, puede realizar las siguientes operaciones:

- Anular una configuración existente: proporcione el mismo parámetro de configuración en la `StartJobRun` solicitud con sus valores de anulación.
- Añadir una configuración adicional: añada el nuevo parámetro de configuración a la `StartJobRun` solicitud con los valores que desee especificar.
- Eliminar una configuración existente: para eliminar una configuración en tiempo de ejecución de una aplicación, proporcione la clave de la configuración que desee eliminar y pase una declaración vacía `{}` para la configuración. No recomendamos eliminar ninguna clasificación que contenga parámetros necesarios para la ejecución de un trabajo. Por ejemplo, si intenta eliminar las propiedades necesarias para un trabajo de Hive, el trabajo fallará.

  Para eliminar una configuración de supervisión de aplicaciones, utilice el método adecuado para el tipo de configuración correspondiente:

  - **`cloudWatchLoggingConfiguration`**- Para eliminarla`cloudWatchLogging`, pase la marca de activación como`false`.
  - **`managedPersistenceMonitoringConfiguration`**- Para eliminar la configuración de persistencia gestionada y volver al estado activado por defecto, introduzca una declaración vacía `{}` para la configuración.
  - **`s3MonitoringConfiguration`**- Para `s3MonitoringConfiguration` eliminarla, pase una declaración vacía `{}` para la configuración.

Ejemplo de anulación

El siguiente ejemplo muestra diferentes operaciones que puede realizar durante el envío de un trabajo en`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
            }
        }
    }'
```

En el momento de la ejecución del trabajo, se aplicarán las siguientes clasificaciones y configuraciones en función de la clasificación de anulación de prioridades descrita en Parámetro de anulación de la configuración de Hive y. Parámetro de anulación de la configuración de Spark

- La clasificación se `spark-defaults` actualizará con las propiedades especificadas a nivel de puesto. Solo se `StartJobRun` considerarán para esta clasificación las propiedades incluidas en ella.
- La clasificación se `spark-executor-log4j2` añadirá a la lista de clasificaciones existente.
- Se eliminará `spark-driver-log4j2` la clasificación.
- Las configuraciones de se `managedPersistenceMonitoringConfiguration` actualizarán con las configuraciones a nivel de trabajo.
- Se `s3MonitoringConfiguration` eliminarán las configuraciones de.
- Las configuraciones de se `cloudWatchLoggingConfiguration` agregarán a las configuraciones de monitoreo existentes.

Personalización de una imagen EMR sin servidor

A partir de Amazon EMR 6.9.0, puede usar imágenes personalizadas para empaquetar las dependencias de las aplicaciones y los entornos de ejecución en un único contenedor con Amazon EMR Serverless. Esto simplifica la forma en que administra las dependencias de la carga de trabajo y hace que sus paquetes sean más portátiles. Al personalizar la imagen EMR sin servidor, se obtienen las siguientes ventajas:

- Instala y configura paquetes optimizados para sus cargas de trabajo. Es posible que estos paquetes no estén ampliamente disponibles en la distribución pública de los entornos de EMR ejecución de Amazon.
- Integra EMR Serverless con los procesos de creación, prueba e implementación establecidos actualmente en su organización, incluidos el desarrollo y las pruebas locales.
- Aplica procesos de seguridad establecidos, como el escaneo de imágenes, que cumplen con los requisitos de cumplimiento y gobierno de su organización.
- Le permite usar sus propias versiones JDK de Python para sus aplicaciones.

EMRServerless proporciona imágenes que puede utilizar como base al crear sus propias imágenes. La imagen base proporciona los contenedores, la configuración y las bibliotecas esenciales para

que la imagen interactúe con EMR Serverless. Puedes encontrar la imagen base en la Amazon ECR Public Gallery. Usa la imagen que coincida con el tipo de aplicación (Spark o Hive) y la versión de lanzamiento. Por ejemplo, si crea una aplicación en la EMR versión 6.9.0 de Amazon, utilice las siguientes imágenes.

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

Requisitos previos

Antes de crear una imagen personalizada EMR sin servidor, complete estos requisitos previos.

1. Crea un ECR repositorio de Amazon en el mismo Región de AWS que utilizas para lanzar aplicaciones EMR sin servidor. Para crear un repositorio ECR privado de Amazon, consulta Crear un repositorio privado.
2. Para conceder a los usuarios acceso a tu ECR repositorio de Amazon, añade las siguientes políticas a los usuarios y roles que crean o actualizan aplicaciones EMR sin servidor con imágenes de este repositorio.

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

Para ver más ejemplos de políticas basadas en la ECR identidad de Amazon, consulte Ejemplos de políticas basadas en la identidad de Amazon Elastic Container Registry.

Paso 1: Cree una imagen personalizada a partir de imágenes base sin servidor EMR

En primer lugar, cree un Dockerfile que comience con una `FROM` instrucción que utilice la imagen base que prefiera. Después de las `FROM` instrucciones, puede incluir cualquier modificación que desee realizar en la imagen. La imagen base establece automáticamente el `USER` `thadoop`. Es posible que esta configuración no tenga permisos para todas las modificaciones que incluya. Como solución alternativa, establezca el valor en`root`, modifique la imagen y, `USER` a continuación, `USER` vuelva a `hadoop:hadoop` configurarlo en. Para ver ejemplos de casos de uso comunes, consulteUso de imágenes personalizadas con 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
```

Una vez que tenga el Dockerfile, cree la imagen con el siguiente comando.

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

Paso 2: Validar la imagen localmente

EMRServerless proporciona una herramienta sin conexión que puede comprobar estáticamente su imagen personalizada para validar los archivos básicos, las variables de entorno y corregir las configuraciones de imagen. Para obtener información sobre cómo instalar y ejecutar la herramienta, consulte la imagen CLI GitHub de Amazon EMR Serverless.

Tras instalar la herramienta, ejecute el siguiente comando para validar una imagen:

```
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
```

Debería ver un resultado similar al siguiente.

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

Paso 3: Sube la imagen a tu ECR repositorio de Amazon

Envía tu ECR imagen de Amazon a tu ECR repositorio de Amazon con los siguientes comandos. Asegúrate de tener los IAM permisos correctos para enviar la imagen a tu repositorio. Para obtener más información, consulta Cómo insertar una imagen en la Guía del ECR usuario de 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
```

Paso 4: Crea o actualiza una aplicación con imágenes personalizadas

Elija el icono AWS Management Console pestaña o AWS CLI separe la pestaña según la forma en que desee iniciar la aplicación y, a continuación, complete los siguientes pasos.

Console

1. Inicia sesión en la consola de EMR Studio en https://console.aws.amazon.com/emr. Navegue hasta su aplicación o cree una nueva aplicación siguiendo las instrucciones de Crear una aplicación.
2. Para especificar imágenes personalizadas al crear o actualizar una aplicación EMR sin servidor, seleccione Configuración personalizada en las opciones de configuración de la aplicación.
3. En la sección Configuración de imagen personalizada, active la casilla de verificación Usar la imagen personalizada con esta aplicación.
4. Pega la ECR imagen de Amazon URI en el URI campo Imagen. EMRServerless usa esta imagen para todos los tipos de trabajadores de la aplicación. Como alternativa, puedes elegir diferentes imágenes personalizadas y pegar diferentes ECR imágenes de Amazon URIs para cada tipo de trabajador.

CLI

- Cree una aplicación con el `image-configuration` parámetro. EMRServerless aplica esta configuración a todos los tipos de trabajadores.

```
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 crear una aplicación con diferentes ajustes de imagen para cada tipo de trabajador, utilice el `worker-type-specifications` parámetro.

```
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 actualizar una aplicación, utilice el `image-configuration` parámetro. EMRServerless aplica esta configuración a todos los tipos de trabajadores.

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

Paso 5: Permita que EMR Serverless acceda al repositorio de imágenes personalizado

Añada la siguiente política de recursos al ECR repositorio de Amazon para permitir que el director del servicio EMR Serverless utilice las `download` solicitudes `getdescribe`, y de este repositorio.

```
{
  "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 práctica recomendada de seguridad, añada una clave de `aws:SourceArn` condición a la política del repositorio. La clave de condición IAM global `aws:SourceArn` garantiza que EMR Serverless utilice el repositorio solo para una aplicaciónARN. Para obtener más información sobre las políticas de ECR repositorios de Amazon, consulta Cómo crear un repositorio privado.

Consideraciones y limitaciones

Cuando trabaje con imágenes personalizadas, tenga en cuenta lo siguiente:

- Utilice la imagen base correcta que coincida con el tipo (Spark o Hive) y la etiqueta de publicación (por ejemplo`emr-6.9.0`) de su aplicación.
- EMRServerless ignora nuestras `[CMD]` `[ENTRYPOINT]` instrucciones del archivo Docker. Utilice las instrucciones habituales del archivo Docker, como`[COPY]`, y. `[RUN]` `[WORKDIR]`
- No debe modificar las variables de entorno `JAVA_HOME`SPARK_HOME,`HIVE_HOME`, `TEZ_HOME` al crear una imagen personalizada.
- Las imágenes personalizadas no pueden superar los 5 GB de tamaño.
- Si modificas archivos binarios o tarros en las imágenes EMR base de Amazon, es posible que se produzcan errores al iniciar la aplicación o el trabajo.

- El ECR repositorio de Amazon debe estar en el mismo Región de AWS que utilizas para lanzar aplicaciones EMR sin servidor.

Configurar VPC el acceso

Puede configurar las aplicaciones EMR sin servidor para que se conecten a sus almacenes de datosVPC, como los clústeres de Amazon Redshift, las bases de datos de Amazon o los VPC buckets de RDS Amazon S3 con puntos de enlace. Su aplicación EMR sin servidor tiene conectividad saliente con los almacenes de datos que contiene. VPC De forma predeterminada, EMR Serverless bloquea el acceso entrante a sus aplicaciones para mejorar la seguridad.

> **Note**
>
> Debe configurar el VPC acceso si quiere utilizar una base de datos externa de Hive metastore para su aplicación. Para obtener información sobre cómo configurar un metaalmacén de Hive externo, consulte Configuración de Metastore.

Crear aplicación

En la página Crear aplicación, puede elegir una configuración personalizada y especificar las subredes y los VPC grupos de seguridad que pueden usar las aplicaciones EMR sin servidor.

VPCs

Elija el nombre de la nube privada virtual (VPC) que contiene sus almacenes de datos. La página de creación de aplicaciones muestra todas VPCs las que haya elegido Región de AWS.

Subredes

Elija las subredes de la VPC que contiene el banco de datos. La página Crear aplicación muestra todas las subredes de los almacenes de datos de su. VPC

Las subredes seleccionadas deben ser subredes privadas. Esto significa que las tablas de enrutamiento asociadas a las subredes no deben tener puertas de enlace de Internet.

Para la conectividad saliente a Internet, las subredes deben tener rutas salientes que utilicen una puerta de enlace. NAT Para configurar una NAT puerta de enlace, consulte Trabajar con puertas de enlace. NAT

Para la conectividad de Amazon S3, las subredes deben tener una NAT puerta de enlace o un VPC punto final configurado. Para configurar un punto de enlace de S3, consulte Crear un VPC punto de enlace de puerta de enlace.

Para obtener conectividad con otros Servicios de AWS fuera deVPC, como Amazon DynamoDB, debe configurar puntos de enlace VPC o una puerta de enlace. NAT Para configurar los puntos de enlace para VPC Servicios de AWS, consulte Trabajar con puntos VPC finales.

Los trabajadores pueden conectarse a sus almacenes de datos VPC a través del tráfico saliente. De forma predeterminada, EMR Serverless bloquea el acceso entrante a los trabajadores para mejorar la seguridad.

Cuando usas AWS Config, EMR Serverless crea un registro de elementos de la interfaz de red elástica para cada trabajador. Para evitar los costos relacionados con este recurso, considere la posibilidad de desactivar `AWS::EC2::NetworkInterface` AWS Config.

> **Note**
>
> Le recomendamos que seleccione varias subredes en varias zonas de disponibilidad. Esto se debe a que las subredes que elija determinan las zonas de disponibilidad disponibles para el lanzamiento de una aplicación EMR sin servidor. Cada trabajador consumirá una dirección IP de la subred en la que se lance. Asegúrese de que las subredes especificadas tengan direcciones IP suficientes para la cantidad de trabajadores que planea lanzar. Para obtener más información sobre la planificación de subredes, consulte. the section called “Prácticas recomendadas para la planificación de subredes”

Grupos de seguridad

Elija uno o más grupos de seguridad que puedan comunicarse con sus almacenes de datos. La página Crear aplicación muestra todos los grupos de seguridad de suVPC. EMRServerless asocia estos grupos de seguridad a las interfaces de red elásticas que están conectadas a las VPC subredes.

> **Note**
>
> Se recomienda crear un grupo de seguridad independiente para las aplicaciones EMR sin servidor. Esto hace que el aislamiento y la administración de las reglas de red sean más

eficientes. Por ejemplo, para comunicarse con los clústeres de Amazon Redshift, puede definir las reglas de tráfico entre los grupos de seguridad Redshift y EMR Serverless, como se muestra en el siguiente ejemplo.

Example Ejemplo: comunicación con clústeres de Amazon Redshift

1. Agregue una regla para el tráfico entrante al grupo de seguridad de Amazon Redshift desde uno de EMR los grupos de seguridad sin servidor.

Tipo	Protocolo	Intervalo de puertos	Origen
Todos TCP	TCP	5439	`emr-serverless-security-group`

2. Agregue una regla para el tráfico saliente desde uno de los grupos de seguridad EMR sin servidor. Puede hacerlo de dos formas. En primer lugar, puede abrir el tráfico saliente a todos los puertos.

Tipo	Protocolo	Rango de puerto	Destino
Todo el tráfico	TCP	ALL	0.0.0.0/0

Como alternativa, puede restringir el tráfico saliente a los clústeres de Amazon Redshift. Esto solo resulta útil cuando la aplicación debe comunicarse con los clústeres de Amazon Redshift y nada más.

Tipo	Protocolo	Intervalo de puertos	Origen
¿Todos TCP	TCP	5439	`redshift-security-group`

Configurar la aplicación

Puede cambiar la configuración de red de una aplicación EMR sin servidor existente desde la página Configurar la aplicación.

Vea los detalles de la ejecución del trabajo

En la página de detalles de la ejecución del trabajo, puede ver la subred utilizada por el trabajo para una ejecución específica. Tenga en cuenta que un trabajo solo se ejecuta en una subred seleccionada de las subredes especificadas.

Prácticas recomendadas para la planificación de subredes

AWS los recursos se crean en una subred que es un subconjunto de direcciones IP disponibles en Amazon. VPC Por ejemplo, una VPC con una máscara de red /16 tiene hasta 65.536 direcciones IP disponibles que se pueden dividir en varias redes más pequeñas mediante máscaras de subred. Por ejemplo, puede dividir este rango en dos subredes, cada una con una máscara /17 y 32.768 direcciones IP disponibles. Una subred reside dentro de una zona de disponibilidad y no puede abarcar varias zonas.

Las subredes deben diseñarse teniendo en cuenta los límites de escalado de las aplicaciones EMR sin servidor. Por ejemplo, si tiene una aplicación que solicita 4 vCpu trabajadores y puede ampliarse hasta 4000vCpu, su aplicación necesitará como máximo 1000 trabajadores para un total de 1000 interfaces de red. Se recomienda crear subredes en varias zonas de disponibilidad. Esto permite a EMR Serverless volver a intentar su trabajo o aprovisionar la capacidad preinicializada en una zona de disponibilidad diferente en el improbable caso de que se produzca un error en una zona de disponibilidad. Por lo tanto, cada subred de al menos dos zonas de disponibilidad debe tener más de 1000 direcciones IP disponibles.

Necesita subredes con un tamaño de máscara inferior o igual a 22 para aprovisionar 1000 interfaces de red. Cualquier máscara superior a 22 no cumplirá el requisito. Por ejemplo, una máscara de subred de /23 proporciona 512 direcciones IP, mientras que una máscara de /22 proporciona 1024 y una máscara de /21 proporciona 2048 direcciones IP. A continuación, se muestra un ejemplo de 4 subredes con una máscara /22 en una máscara de red VPC de /16 que se pueden asignar a distintas zonas de disponibilidad. Hay una diferencia de cinco entre las direcciones IP disponibles y utilizables porque las cuatro primeras direcciones IP y la última dirección IP de cada subred están reservadas por AWS.

ID de subred	Dirección de subred	Máscara de subred	Rangos de direcciones IP	Direcciones IP disponibles	Direcciones IP utilizables
1	10.0.0.0	255,255,252,0/22	10,0,0 - 10,03,255	1 024	1.019
2	10.0.4.0	255,255,252,0/22	10,04,0 - 10,07255	1 024	1.019
3	10.0.8.0	255,255,252,0/22	10,04,0 - 10,07255	1 024	1.019
4	10.0.12.0	255,255,252,0/22	10,012,0 - 10,0,15255	1 024	1.019

Debe evaluar si su carga de trabajo es la más adecuada para trabajadores de mayor tamaño. El uso de trabajadores de mayor tamaño requiere menos interfaces de red. Por ejemplo, el uso de 16 vCpu trabajadores con un límite de escalado de aplicaciones de 4000 vCpu requerirá como máximo 250 trabajadores para un total de 250 direcciones IP disponibles para aprovisionar las interfaces de red. Necesita subredes en varias zonas de disponibilidad con un tamaño de máscara inferior o igual a 24 para aprovisionar 250 interfaces de red. Cualquier tamaño de máscara superior a 24 ofrece menos de 250 direcciones IP.

Si comparte subredes entre varias aplicaciones, cada subred debe diseñarse teniendo en cuenta los límites de escalamiento colectivos de todas sus aplicaciones. Por ejemplo, si tiene 3 aplicaciones que solicitan 4 vCpu trabajadores y cada una puede ampliarse hasta 4000 vCpu con una cuota de 12 000 basada en el servicio a vCpu nivel de cuenta, cada subred necesitará 3000 direcciones IP disponibles. Si la VPC que desea utilizar no tiene un número suficiente de direcciones IP, intente aumentar el número de direcciones IP disponibles. Para ello, asocie bloques adicionales de enrutamiento entre dominios sin clase (CIDR) a su. VPC Para obtener más información, consulta Asociar IPv4 CIDR bloques adicionales a los tuyos VPC en la Guía del VPC usuario de Amazon.

Puede utilizar una de las muchas herramientas disponibles en línea para generar rápidamente definiciones de subredes y revisar su rango disponible de direcciones IP.

Opciones de arquitectura Amazon EMR Serverless

La arquitectura del conjunto de instrucciones de la aplicación Amazon EMR Serverless determina el tipo de procesadores que la aplicación utiliza para ejecutar el trabajo. Amazon EMR ofrece dos opciones de arquitectura para su aplicación: x86_64 y arm64. EMRServerless se actualiza automáticamente a la última generación de instancias a medida que están disponibles, por lo que sus aplicaciones pueden usar las instancias más nuevas sin que usted deba esforzarse más.

Temas

Uso de la arquitectura x86_64

La arquitectura x86_64 también se conoce como x86 de 64 bits o x64. x86_64 es la opción predeterminada para las aplicaciones sin servidor. EMR Esta arquitectura utiliza procesadores basados en x86 y es compatible con la mayoría de las herramientas y bibliotecas de terceros.

La mayoría de las aplicaciones son compatibles con la plataforma de hardware x86 y pueden ejecutarse correctamente en la arquitectura x86_64 predeterminada. Sin embargo, si su aplicación es compatible con 64 bitsARM, puede cambiarse a arm64 para utilizar los procesadores Graviton y así mejorar el rendimiento, la potencia de cálculo y la memoria. Ejecutar instancias en una arquitectura arm64 cuesta menos que ejecutar instancias del mismo tamaño en una arquitectura x86.

Uso de la arquitectura arm64 (Graviton)

AWS Los procesadores Graviton están diseñados a medida por AWS con núcleos ARM Neoverse de 64 bits y aprovechan la arquitectura arm64 (también conocida como Arch64 o 64 bits). ARM La AWS La línea de procesadores Graviton disponible en EMR Serverless incluye los procesadores Graviton3 y Graviton2. Estos procesadores ofrecen una relación precio-rendimiento superior para las cargas de trabajo Spark y Hive en comparación con las cargas de trabajo equivalentes que se ejecutan en la arquitectura x86_64. EMRServerless utiliza automáticamente la última generación de procesadores cuando está disponible sin que usted se esfuerce por actualizarlos a la última generación de procesadores.

Lanzamiento de nuevas aplicaciones compatibles con Graviton

Utilice uno de los métodos siguientes para iniciar una aplicación que utilice la arquitectura arm64.

AWS CLI

Para iniciar una aplicación con procesadores Graviton desde AWS CLI, especifique `ARM64` como `architecture` parámetro en. `create-application` API Proporcione los valores adecuados para su aplicación en los demás parámetros.

```
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 una aplicación con los procesadores Graviton de EMR Studio, elija arm64 como opción de arquitectura al crear o actualizar una aplicación.

Configurar las aplicaciones existentes para usar Graviton

Puede configurar sus aplicaciones Amazon EMR Serverless existentes para que utilicen la arquitectura Graviton (arm64) con, SDK AWS CLI, o Studio. EMR

Para convertir una aplicación existente de x86 a arm64

1. Confirme que está utilizando la última versión principal de AWS CLI/SDKque sea compatible con el `architecture` parámetro.
2. Confirme que no hay ningún trabajo en ejecución y, a continuación, detenga la aplicación.

   ```
   aws emr-serverless stop-application \
    --application-id application-id \
    --region us-west-2
   ```
3. Para actualizar la aplicación para que utilice Graviton, especifique `ARM64` el `architecture` parámetro en. `update-application` API

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

4. Para comprobar que la CPU arquitectura de la aplicación es actualARM64, utilice la `get-application`API.

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

5. Cuando esté listo, reinicie la aplicación.

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

Consideraciones a la hora de utilizar Graviton

Antes de iniciar una aplicación EMR sin servidor con arm64 compatible con Graviton, confirme lo siguiente.

Compatibilidad con bibliotecas

Al seleccionar Graviton (arm64) como opción de arquitectura, asegúrese de que los paquetes y bibliotecas de terceros sean compatibles con la arquitectura de 64 bits. ARM Para obtener información sobre cómo empaquetar las bibliotecas de Python en un entorno virtual de Python que sea compatible con la arquitectura seleccionada, consulteUso de bibliotecas de Python con EMR Serverless.

Para obtener más información sobre cómo configurar una carga de trabajo de Spark o Hive para que utilice 64 bitsARM, consulta la AWS El repositorio Graviton Getting Started está en. GitHub Este repositorio contiene recursos esenciales que pueden ayudarlo a comenzar con el Graviton ARM basado en Graviton.

Introduzca datos en S3 Express One Zone con EMR Serverless

Con las EMR versiones 7.2.0 y posteriores de Amazon, puede usar EMR Serverless con la clase de almacenamiento Amazon S3 Express One Zone para mejorar el rendimiento al ejecutar tareas y cargas de trabajo. S3 Express One Zone es una clase de almacenamiento Amazon S3 de zona única y alto rendimiento que ofrece un acceso uniforme a los datos en milisegundos de un solo dígito para la mayoría de las aplicaciones sensibles a la latencia. En el momento de su lanzamiento, S3 Express One Zone ofrece el almacenamiento de objetos en la nube con la latencia más baja y el rendimiento más alto de Amazon S3.

Requisitos previos

- Permisos de S3 Express One Zone: cuando S3 Express One Zone realiza inicialmente una acción`GET`, como un objeto de S3 o `PUT` sobre un objeto de S3`LIST`, la clase de almacenamiento llama en tu nombre. `CreateSession` Su IAM política debe permitir el `s3express:CreateSession` permiso para que S3A el conector puede invocar el `CreateSession`API. Para ver un ejemplo de política con ese permiso, consulte Introducción a S3 Express One Zone.
- S3A conector: para configurar Spark para que acceda a los datos de un depósito de Amazon S3 que utilice la clase de almacenamiento S3 Express One Zone, debe utilizar el conector Apache Hadoop S3A. Para usar el conector, asegúrese de que todos los S3 URIs utilicen el `s3a` esquema. Si no es así, puede cambiar la implementación del sistema de archivos que utiliza para los esquemas `s3` y `s3n`.

Para cambiar el esquema `s3`, especifique las siguientes configuraciones de clúster:

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

Para cambiar el esquema `s3n`, especifique las siguientes configuraciones de clúster:

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

Introducción a S3 Express One Zone

Siga estos pasos para empezar a utilizar S3 Express One Zone.

1. Cree un VPC punto final. Añada el punto final `com.amazonaws.us-west-2.s3express` al VPC punto final.
2. Siga Introducción a Amazon EMR Serverless para crear una aplicación con la etiqueta de EMR lanzamiento de Amazon 7.2.0 o superior.
3. Configure su aplicación para que utilice el VPC punto final recién creado, un grupo de subredes privadas y un grupo de seguridad.
4. Añada el `CreateSession` permiso a su función de ejecución de tareas.

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

5. Dirija su trabajo. Tenga en cuenta que debe usar el `S3A` esquema para acceder a los depósitos de S3 Express One Zone.

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


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

Trabajos en ejecución

Después de aprovisionar la solicitud, puede enviar los trabajos a la solicitud. En esta sección se explica cómo utilizar el AWS CLI para ejecutar estos trabajos. En esta sección también se identifican los valores predeterminados para cada tipo de aplicación disponible en EMR Serverless.

Temas

Estados de ejecuciones de trabajos

Cuando envía una ejecución de tareas a una cola de tareas de Amazon EMR Serverless, la ejecución de tareas pasa a ese estado. `SUBMITTED` El estado de un trabajo pasa desde el `SUBMITTED` final `RUNNING` hasta que llega a `FAILEDSUCCESS`, o. `CANCELLING`

Las ejecuciones de trabajos pueden tener los siguientes estados:

Estado	Descripción
Presentado	El estado inicial del trabajo al enviar un trabajo se ejecuta en EMR Serverless. El trabajo espera a que se programe para la solicitud . EMRServerless comienza a priorizar y programar la ejecución del trabajo.

Estado	Descripción
Pendiente	El programador evalúa la ejecución del trabajo para priorizar y programar la ejecución de la aplicación.
Programado	EMRServerless ha programado la ejecución del trabajo para la aplicación y está asignando recursos para ejecutarlo.
En ejecución	EMRServerless ha asignado los recursos que el trabajo necesita inicialmente y el trabajo se está ejecutando en la aplicación. En las aplicaciones de Spark, esto significa que el estado del proceso del controlador de Spark es `running`.
Con error	EMRServerless no pudo enviar el trabajo ejecutado a la aplicación o se completó sin éxito. Consulte `StateDetails` para obtener información adicional acerca de este error en el trabajo.
Correcto	La ejecución del trabajo se ha completado correctamente.
Cancelling	`CancelJobRun` APIHa solicitado la cancelación de la ejecución de la tarea o se ha agotado el tiempo de espera de la ejecución de la tarea. EMRServerless está intentando cancelar el trabajo en la aplicación y liberar los recursos.
Cancelado	La ejecución del trabajo se canceló correctamente y se liberaron los recursos que utilizaba.

Ejecución de trabajos desde la consola de EMR Studio

Puede enviar las ejecuciones de tareas a aplicaciones EMR sin servidor y ver las tareas desde la consola de EMR Studio. Para crear una aplicación EMR sin servidor o acceder a ella en la consola de EMR Studio, sigue las instrucciones de Cómo empezar desde la consola.

Enviar un trabajo

En la página Enviar trabajo, puedes enviar un trabajo a una aplicación EMR sin servidor de la siguiente manera.

Spark

1. En el campo Nombre, introduzca un nombre para la ejecución del trabajo.
2. En el campo Función de ejecución, introduzca el nombre de la IAM función que la aplicación EMR sin servidor puede asumir para la ejecución de la tarea. Para obtener más información sobre las funciones en tiempo de ejecución, consulteFunciones de tiempo de ejecución de trabajos para Amazon EMR Serverless.
3. En el campo Ubicación del script, introduzca la ubicación de Amazon S3 del script o JAR que desee ejecutar. Para los trabajos de Spark, el script puede ser un archivo Python (`.py`) o un archivo JAR (`.jar`).
4. Si la ubicación del script es un JAR archivo, introduce el nombre de la clase que es el punto de entrada del trabajo en el campo Clase principal.
5. (Opcional) Introduzca valores para los campos restantes.

   - Argumentos del script: introduzca los argumentos que desee pasar a su script principal JAR o de Python. El código lee estos parámetros. Separe cada argumento de la matriz con una coma.
   - Propiedades de Spark: expande la sección de propiedades de Spark e introduce cualquier parámetro de configuración de Spark en este campo.

   > **Note**
   >
   > Si especificas los tamaños del controlador y del ejecutor de Spark, debes tener en cuenta la sobrecarga de memoria. Especifique los valores de sobrecarga de memoria en las propiedades `spark.driver.memoryOverhead` y`spark.executor.memoryOverhead`. La sobrecarga de memoria tiene un valor

predeterminado del 10% de la memoria del contenedor, con un mínimo de 384 MB. La memoria ejecutora y la sobrecarga de memoria juntas no pueden superar la memoria de trabajo. Por ejemplo, el máximo de un `spark.executor.memory` dispositivo de trabajo de 30 GB debe ser de 27 GB.

- Configuración de trabajo: especifique cualquier configuración de trabajo en este campo. Puede usar estas configuraciones de trabajo para anular las configuraciones predeterminadas de las aplicaciones.
- Ajustes adicionales: activa o desactiva la AWS Glue Data Catalog como un metaalmacén y modifica la configuración del registro de la aplicación. Para obtener más información sobre las configuraciones del metaalmacén, consulte. Configuración de Metastore Para obtener más información sobre las opciones de registro de aplicaciones, consulteAlmacenamiento de registros.
- Etiquetas: asigne etiquetas personalizadas a la aplicación.

6. Seleccione Enviar el trabajo.

Hive

1. En el campo Nombre, introduzca un nombre para la ejecución del trabajo.
2. En el campo Función de ejecución, introduzca el nombre de la IAM función que la aplicación EMR sin servidor puede asumir para la ejecución de la tarea.
3. En el campo Ubicación del script, introduzca la ubicación de Amazon S3 del script o JAR que desee ejecutar. Para los trabajos de Hive, el script debe ser un archivo Hive (`.sql`).
4. (Opcional) Introduzca valores para los campos restantes.
   - Ubicación del script de inicialización: introduzca la ubicación del script que inicializa las tablas antes de que se ejecute el script de Hive.
   - Propiedades de la colmena: amplíe la sección de propiedades de la colmena e introduzca los parámetros de configuración de la colmena en este campo.
   - Configuración de trabajo: especifique cualquier configuración de trabajo. Puede utilizar estas configuraciones de trabajo para anular las configuraciones predeterminadas de las aplicaciones. Para los trabajos de Hive, `hive.exec.scratchdir` y `hive.metastore.warehouse.dir` son propiedades obligatorias en la `hive-site` configuración.

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

- Ajustes adicionales: active o desactive el AWS Glue Data Catalog como un metaalmacén y modifica la configuración del registro de la aplicación. Para obtener más información sobre las configuraciones del metaalmacén, consulte. Configuración de Metastore Para obtener más información sobre las opciones de registro de aplicaciones, consulteAlmacenamiento de registros.
- Etiquetas: asigne cualquier etiqueta personalizada a la aplicación.

5. Seleccione Enviar el trabajo.

Vista de las ejecuciones de trabajo

En la pestaña Ejecuciones de tareas de la página de detalles de una aplicación, puede ver las ejecuciones de tareas y realizar las siguientes acciones para las ejecuciones de tareas.

Cancelar trabajo: para cancelar una ejecución de trabajo que está en ese `RUNNING` estado, elija esta opción. Para obtener más información sobre las transiciones de ejecución de tareas, consulteEstados de ejecuciones de trabajos.

Clonar un trabajo: para clonar un trabajo anterior ejecutado y volver a enviarlo, seleccione esta opción.

Ejecutar trabajos desde AWS CLI

Puede crear, describir y eliminar trabajos individuales en AWS CLI. También puedes hacer una lista de todos tus trabajos para verlos de un vistazo.

Para enviar un nuevo trabajo, utilice`start-job-run`. Proporcione el ID de la aplicación que desea ejecutar, junto con las propiedades específicas del trabajo. Para ver ejemplos de Spark, consulte. Empleos en Spark Para ver ejemplos de Hive, consulteEmpleos en Hive. Este comando devuelve su `application-id`ARN, y una nueva`job-id`.

Cada ejecución de trabajo tiene un tiempo de espera establecido. Si la ejecución del trabajo supera esta duración, EMR Serverless la cancelará automáticamente. El tiempo de espera predeterminado es de 12 horas. Al iniciar la ejecución del trabajo, puede configurar esta configuración de tiempo de espera en un valor que cumpla con los requisitos del trabajo. Configure el valor con la `executionTimeoutMinutes` propiedad.

```
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 describir un trabajo, utilice`get-job-run`. Este comando devuelve las configuraciones específicas del trabajo y la capacidad establecida para el nuevo trabajo.

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

```
--job-run-id job-id \
--application-id application-id
```

Para enumerar sus trabajos, utilice. `list-job-runs` Este comando devuelve un conjunto abreviado de propiedades que incluye el tipo de trabajo, el estado y otros atributos de alto nivel. Si no desea ver todos sus trabajos, puede especificar el número máximo de trabajos que desea ver, hasta 50. En el ejemplo siguiente se especifica que desea ver las dos últimas ejecuciones de sus trabajos.

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

Para cancelar un trabajo, utilice`cancel-job-run`. Proporcione el `application-id` y el `job-id` del trabajo que desea cancelar.

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

Para obtener más información sobre cómo ejecutar trabajos desde la AWS CLI, consulte la APIReferencia sobre EMR sistemas sin servidor.

Uso de discos optimizados para reproducción aleatoria

Con las EMR versiones 7.1.0 y posteriores de Amazon, puede usar discos optimizados para la reproducción aleatoria cuando ejecute tareas de Apache Spark o Hive para mejorar el rendimiento de las cargas de trabajo con un uso intensivo de E/S. En comparación con los discos estándar, los discos optimizados para la reproducción aleatoria ofrecen una mayor potencia IOPS (operaciones de E/S por segundo) para acelerar el movimiento de los datos y reducir la latencia durante las operaciones de mezcla aleatoria. Los discos optimizados para la reproducción aleatoria le permiten conectar discos de hasta 2 TB por trabajador, lo que le permite configurar la capacidad adecuada para sus requisitos de carga de trabajo.

Ventajas principales

Los discos optimizados para Shuffle ofrecen las siguientes ventajas.

- Alto IOPS rendimiento: los discos optimizados para la reproducción aleatoria ofrecen una calidad superior a la de los discos estándar, IOPS lo que permite una reorganización de datos más rápida

y eficiente durante las tareas de Spark y Hive y otras cargas de trabajo que requieren un uso intensivo de la mezcla.

- Tamaño de disco más grande: los discos optimizados para Shuffle admiten tamaños de disco de 20 GB a 2 TB por trabajador, por lo que puede elegir la capacidad adecuada en función de sus cargas de trabajo.

Introducción

Consulte los siguientes pasos para usar discos optimizados para la reproducción aleatoria en sus flujos de trabajo.

Spark

1. Cree una aplicación de la versión 7.1.0 EMR sin servidor con el siguiente comando.

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

2. Configura tu trabajo de Spark para que incluya los parámetros `spark.emr-serverless.driver.disk.type` o para que se ejecute con discos `spark.emr-serverless.executor.disk.type` optimizados para la reproducción aleatoria. Puedes usar uno o ambos parámetros, según tu 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 obtener más información, consulta las propiedades de los trabajos de Spark.

Hive

1. Crea una aplicación de la versión 7.1.0 EMR sin servidor con el siguiente comando.

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

2. Configure su trabajo de Hive para que incluya los parámetros `hive.driver.disk.type` o para que se ejecute con discos optimizados `hive.tez.disk.type` para la reproducción aleatoria. Puede usar uno o ambos parámetros, según su 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 obtener más información, consulta Hive job properties.

Configuración de una aplicación con capacidad preinicializada

Consulte los siguientes ejemplos para crear aplicaciones basadas en la EMR versión 7.1.0 de Amazon. Estas aplicaciones tienen las siguientes propiedades:

- 5 controladores Spark preinicializados, cada uno con 2 VCPU, 4 GB de memoria y 50 GB de disco optimizado para reproducción aleatoria.
- 50 ejecutores preinicializados, cada uno con 4 VCPU, 8 GB de memoria y 500 GB de disco optimizado para reproducción aleatoria.

Cuando esta aplicación ejecuta tareas de Spark, primero consume los trabajadores preinicializados y, a continuación, amplía los trabajadores bajo demanda hasta una capacidad máxima de 400 v y 1024 GB de memoria. CPU Si lo desea, puede omitir la capacidad de una u otra opción. `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"
  }'
```

Trabajos de streaming

Un trabajo de streaming en EMR Serverless es un modo de trabajo que permite analizar y procesar los datos de streaming prácticamente en tiempo real. Estos trabajos de larga duración recopilan los datos de streaming y procesan los resultados de forma continua a medida que llegan los datos. Los trabajos de streaming son los más adecuados para las tareas que requieren un procesamiento de datos en tiempo real, como los análisis prácticamente en tiempo real, la detección de fraudes y los motores de recomendaciones. EMRLos trabajos de streaming sin servidor proporcionan optimizaciones, como la resiliencia de los trabajos integrada, la supervisión en tiempo real, la mejora de la gestión de registros y la integración con conectores de streaming.

A continuación se muestran algunos casos de uso de los trabajos de streaming:

- Análisis casi en tiempo real: los trabajos de streaming en Amazon EMR Serverless le permiten procesar datos de streaming prácticamente en tiempo real, de modo que puede realizar análisis en tiempo real de flujos de datos continuos, como datos de registro, datos de sensores o datos de secuencias de clics, a fin de obtener información y tomar decisiones oportunas en función de la información más reciente.
- Detección de fraudes: puede utilizar los trabajos de streaming para detectar fraudes prácticamente en tiempo real en transacciones financieras, operaciones con tarjetas de crédito o actividades en línea al analizar los flujos de datos e identificar patrones o anomalías sospechosos a medida que se producen.
- Motores de recomendaciones: los trabajos de streaming pueden procesar los datos de actividad de los usuarios y actualizar los modelos de recomendaciones. Esto abre la posibilidad de realizar recomendaciones personalizadas y en tiempo real en función de los comportamientos y las preferencias.
- Análisis de redes sociales: los trabajos de streaming pueden procesar datos de las redes sociales, como tuits, comentarios y publicaciones, para que las organizaciones puedan monitorear las tendencias, analizar las opiniones y gestionar la reputación de la marca casi en tiempo real.
- Análisis de Internet de las cosas (IoT): los trabajos de streaming pueden gestionar y analizar flujos de datos a alta velocidad de dispositivos, sensores y maquinaria conectada de IoT, de modo que puede ejecutar la detección de anomalías, el mantenimiento predictivo y otros casos de uso de análisis de IoT.
- Análisis del flujo de clics: los trabajos de streaming permiten procesar y analizar los datos del flujo de clics procedentes de sitios web o aplicaciones móviles. Las empresas que utilizan estos datos

pueden realizar análisis para obtener más información sobre el comportamiento de los usuarios, personalizar las experiencias de los usuarios y optimizar las campañas de marketing.

- Supervisión y análisis de registros: los trabajos de streaming también pueden procesar datos de registro de servidores, aplicaciones y dispositivos de red. Esto le permite detectar anomalías, solucionar problemas y mejorar el estado y el rendimiento del sistema.

Ventajas clave

La transmisión de trabajos en EMR Serverless proporciona automáticamente resiliencia laboral, que es una combinación de los siguientes factores:

- Reintento automático: EMR Serverless reintenta automáticamente cualquier trabajo que haya fallado sin ninguna intervención manual por tu parte.
- Resiliencia de la zona de disponibilidad (AZ): EMR Serverless cambia automáticamente los trabajos de streaming a una zona de disponibilidad en buen estado si la zona de disponibilidad original tiene problemas.
- Administración de registros:
  - Rotación de registros: para una administración más eficiente del almacenamiento en disco, EMR Serverless rota periódicamente los registros para trabajos de transmisión prolongados. De este modo, se evita la acumulación de registros que podrían consumir todo el espacio en disco.
  - Compactación de registros: le ayuda a administrar y optimizar de manera eficiente los archivos de registro en sistemas de persistencia gestionada. La compactación también mejora la experiencia de depuración cuando se utiliza el servidor de historial de chispas gestionado.

Fuentes de datos y sumideros de datos compatibles

EMRServerless funciona con varias fuentes de datos de entrada y receptores de datos de salida:

- Fuentes de datos de entrada compatibles: Amazon Kinesis Data Streams, Amazon Managed Streaming for Apache Kafka y clústeres de Apache Kafka autogestionados. De forma predeterminada, las EMR versiones 7.1.0 y posteriores de Amazon incluyen el conector Amazon Kinesis Data Streams, por lo que no es necesario crear ni descargar ningún paquete adicional.
- Receptores de datos de salida compatibles: AWS Tablas de Glue Data Catalog, Amazon S3, Amazon Redshift, MySQL, Postgre SQL Oracle, Oracle, Microsoft, Apache IcebergSQL, Delta Lake y Apache Hudi.

Consideraciones y limitaciones

Cuando utilice trabajos de streaming, tenga en cuenta las siguientes consideraciones y limitaciones.

- Los trabajos de streaming son compatibles con las EMRversiones 7.1.0 y superiores de Amazon.
- EMRServerless espera que los trabajos de streaming se ejecuten durante mucho tiempo, por lo que no se puede establecer un tiempo de espera de ejecución para limitar el tiempo de ejecución del trabajo.
- Los trabajos de streaming solo son compatibles con el motor Spark, que se basa en el marco de streaming estructurado.
- EMRServerless vuelve a intentar los trabajos de streaming de forma indefinida y no puedes personalizar el número máximo de intentos. La función de prevención de errores se incluye automáticamente para detener el reintento del trabajo si la cantidad de intentos fallidos ha superado el umbral establecido durante un período de una hora. El límite predeterminado es de cinco intentos fallidos en una hora. Puede configurar este umbral para que esté comprendido entre 1 y 10 intentos. Para obtener más información, consulte Job Resiliency.
- Los trabajos de transmisión tienen puntos de control para guardar el estado y el progreso del tiempo de ejecución, por lo que EMR Serverless puede reanudar el trabajo de transmisión desde el último punto de control. Para obtener más información, consulte Recuperarse de errores con puntos de control en la documentación de Apache Spark.

Cómo empezar a transmitir trabajos

Consulta las siguientes instrucciones para aprender a empezar a hacer streaming de trabajos.

1. Siga Introducción a Amazon EMR Serverless para crear una aplicación. Tenga en cuenta que su aplicación debe ejecutar la EMRversión 7.1.0 o superior de Amazon.
2. Una vez que la solicitud esté lista, defina el `mode` parámetro `STREAMING` para enviar un trabajo de streaming, de forma similar a la siguiente AWS CLI ejemplo.

```
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 compatibles

Los conectores de transmisión facilitan la lectura de datos de una fuente de transmisión y también pueden escribirlos en un receptor de transmisión.

Los siguientes son los conectores de transmisión compatibles:

Conector Amazon Kinesis Data Streams

El conector Amazon Kinesis Data Streams para Apache Spark permite crear aplicaciones y canalizaciones de streaming que consumen datos de Amazon Kinesis Data Streams y los escriben en ellos. El conector permite aumentar el consumo del ventilador con una velocidad de lectura específica de hasta 2 MB/segundo por fragmento. De forma predeterminada, Amazon EMR Serverless 7.1.0 y versiones posteriores incluyen el conector, por lo que no es necesario compilar ni descargar ningún paquete adicional. Para obtener más información sobre el conector, consulte la spark-sql-kinesis-connector página en. GitHub

A continuación se muestra un ejemplo de cómo iniciar la ejecución de un trabajo con la dependencia del conector de 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 conectarse a Kinesis Data Streams, debe configurar EMR la aplicación Serverless VPC con acceso y usar VPC un punto final para permitir el acceso privado, o usar NAT una puerta de enlace para obtener acceso público. Para obtener más información, consulte Configuración del acceso. VPC También debe asegurarse de que su función de ejecución de tareas tenga los permisos de lectura y escritura necesarios para acceder a los flujos de datos necesarios. Para obtener más información sobre cómo configurar un rol de ejecución de tareas, consulte Funciones de ejecución de tareas para Amazon EMR Serverless. Para obtener una lista completa de todos los permisos necesarios, consulte la spark-sql-kinesis-connector página en GitHub.

Conector Apache Kafka

El conector Apache Kafka para la transmisión estructurada de Spark es un conector de código abierto de la comunidad de Spark y está disponible en un repositorio de Maven. Este conector facilita que las aplicaciones de streaming estructurado de Spark lean y escriban datos en Apache Kafka autogestionados y Amazon Managed Streaming for Apache Kafka. Para obtener más información sobre el conector, consulte la guía de integración entre Structured Streaming y Kafka en la documentación de Apache Spark.

En el siguiente ejemplo, se muestra cómo incluir el conector Kafka en la solicitud de ejecución de un trabajo.

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

```
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>"
    }
}'
```

La versión del conector Apache Kafka depende de la versión de EMR Serverless y de la versión de Spark correspondiente. Para encontrar la versión correcta de Kafka, consulta la Guía de integración entre Structured Streaming y Kafka.

Para utilizar Amazon Managed Streaming for Apache Kafka Kafka con IAM autenticación, debe incluir otra dependencia para permitir que el conector de Kafka se conecte a Amazon con. MSK IAM Para obtener más información, consulte el aws-msk-iam-auth repositorio en. GitHub También debe asegurarse de que el rol de ejecución del trabajo tenga los IAM permisos necesarios. El siguiente ejemplo muestra cómo utilizar el conector con la IAM autenticación.

```
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 utilizar el conector Kafka y la biblioteca de IAM autenticación de Amazon, MSK debe configurar la aplicación EMR Serverless con VPC acceso. Sus subredes deben tener acceso a Internet y utilizar una NAT puerta de enlace para acceder a las dependencias de Maven. Para obtener más información, consulte Configuración del acceso. VPC Las subredes deben tener conectividad de red para acceder al clúster de Kafka. Esto ocurre independientemente de si su clúster de Kafka es autogestionado o si utiliza Amazon Managed Streaming for Apache Kafka Kafka.

Administración de registros de trabajos en streaming

Rotación de registros

Los trabajos de streaming admiten la rotación de registros para los registros de aplicaciones y eventos de Spark. La rotación de registros evita que los trabajos de streaming prolongados generen archivos de registro de gran tamaño que podrían ocupar todo el espacio disponible en el disco. La rotación de registros le ayuda a ahorrar espacio en disco y evita errores en los trabajos debido a la falta de espacio en disco. Para obtener más información, consulte Rotación de registros.

Compactación de troncos

Los trabajos de streaming también admiten la compactación de registros para los registros de eventos de Spark siempre que haya registros gestionados disponibles. Para obtener más información sobre el registro gestionado, consulta Cómo registrar con almacenamiento gestionado. Los trabajos de streaming pueden durar mucho tiempo, y la cantidad de datos de eventos puede acumularse con el tiempo y aumentar considerablemente el tamaño de los archivos de registro. El servidor de historial de Spark lee y carga estos eventos en la memoria para la interfaz de usuario de la aplicación Spark. Este proceso puede provocar latencias y costes elevados, especialmente si los registros de eventos almacenados en Amazon S3 son muy grandes.

La compactación de registros reduce el tamaño del registro de eventos, por lo que el Spark History Server no tiene que cargar más de 1 GB de registros de eventos en ningún momento. Para obtener más información, consulte Supervisión e instrumentación en la documentación de Apache Spark.

Empleos en Spark

Puedes ejecutar trabajos de Spark en una aplicación con el `type` parámetro establecido en`SPARK`. Los trabajos deben ser compatibles con la versión de Spark compatible con la versión de EMR lanzamiento de Amazon. Por ejemplo, cuando ejecuta trabajos con la EMR versión 6.6.0 de Amazon, su trabajo debe ser compatible con Apache Spark 3.2.0. Para obtener información sobre las versiones de la aplicación para cada versión, consulte. Versiones de lanzamiento de Amazon EMR Serverless

Parámetros de trabajo de Spark

Cuando utilizas el `StartJobRun`APIpara ejecutar un trabajo de Spark, puedes especificar los siguientes parámetros.

Parámetros necesarios

- Función de ejecución de trabajos de Spark
- Parámetro del controlador de tareas de Spark
- Parámetro de anulación de la configuración de Spark
- Impulse la optimización dinámica de la asignación de recursos

Función de ejecución de trabajos de Spark

Úselo **`executionRoleArn`**ARNpara especificar el IAM rol que usa su aplicación para ejecutar los trabajos de Spark. Este rol debe contener los siguientes permisos:

- Lea desde los buckets de S3 u otras fuentes de datos en las que residen sus datos
- Lea los buckets o prefijos de S3 donde reside su PySpark script o archivo JAR
- Escriba en los cubos de S3 en los que desee escribir el resultado final
- Escribe los registros en un bucket o prefijo de S3 que especifique `S3MonitoringConfiguration`
- Acceda a KMS las claves si las usa KMS para cifrar los datos de su bucket de S3
- Acceso al AWS Glue Data Catalog si usas Spark SQL

Si tu trabajo de Spark lee o escribe datos en o desde otras fuentes de datos, especifica los permisos adecuados en esta IAM función. Si no le otorgas estos permisos al IAM rol, es posible que el trabajo no funcione. Para obtener más información, consulte Funciones de tiempo de ejecución de trabajos para Amazon EMR Serverless y Almacenamiento de registros.

Parámetro del controlador de tareas de Spark

Se utiliza **`jobDriver`**para proporcionar información al trabajo. El parámetro del controlador de tareas solo acepta un valor para el tipo de trabajo que desee ejecutar. Para un trabajo de Spark, el valor del parámetro es`sparkSubmit`. Puedes usar este tipo de trabajo para ejecutar Scala, Java PySpark, SparkR y cualquier otro trabajo compatible a través de Spark submit. Los trabajos de Spark tienen los siguientes parámetros:

- **`sparkSubmitParameters`**— Estos son los parámetros adicionales de Spark que quieres enviar al trabajo. Usa este parámetro para anular las propiedades predeterminadas de Spark, como la memoria del controlador o el número de ejecutores, como los definidos en los argumentos `--conf` o`--class`.

- **`entryPointArguments`**— Esta es una matriz de argumentos que quieres pasar a tu archivo principal JAR o de Python. Debería manejar la lectura de estos parámetros mediante su código de punto de entrada. Separa cada argumento de la matriz con una coma.
- **`entryPoint`**— Esta es la referencia en Amazon S3 al archivo principal JAR o de Python que desea ejecutar. Si utiliza Scala o JavaJAR, especifique la clase de entrada principal `SparkSubmitParameters` utilizando el `--class` argumento.

Para obtener más información, consulte Launching Applications with spark-submit.

Parámetro de anulación de la configuración de Spark

Se utiliza **`configurationOverrides`**para anular las propiedades de configuración a nivel de supervisión y de aplicación. Este parámetro acepta un JSON objeto con los dos campos siguientes:

- **`monitoringConfiguration`**- Utilice este campo para especificar el Amazon S3 URL (`s3MonitoringConfiguration`) en el que desea que el trabajo EMR sin servidor almacene los registros de su trabajo de Spark. Asegúrate de haber creado este depósito con el mismo Cuenta de AWS que aloja tu aplicación, y en el mismo Región de AWS donde se ejecuta su trabajo.
- **`applicationConfiguration`**— Para anular las configuraciones predeterminadas de las aplicaciones, puede proporcionar un objeto de configuración en este campo. Puede utilizar una sintaxis abreviada para proporcionar la configuración o puede hacer referencia al objeto de configuración en un archivo. JSON Los objetos de configuración se componen de una clasificación, propiedades y configuraciones anidadas opcionales. Las propiedades consisten en las configuraciones que desea anular en ese archivo. Puede especificar varias clasificaciones para varias aplicaciones en un único objeto. JSON

> **Note**
>
> Las clasificaciones de configuración disponibles varían según la versión específica de EMR Serverless. Por ejemplo, las clasificaciones para el Log4j personalizado solo `spark-executor-log4j2` están disponibles en las versiones 6.8.0 `spark-driver-log4j2` y superiores.

Si utilizas la misma configuración en la anulación de una aplicación y en los parámetros de envío de Spark, los parámetros de envío de Spark tienen prioridad. Las configuraciones tienen la siguiente prioridad, de mayor a menor:

- Configuración que proporciona EMR Serverless al crear`SparkSession`.
- Configuración que se proporciona como parte del `sparkSubmitParameters` `--conf` argumento.
- La configuración que se proporciona como parte de la aplicación se anula al iniciar un trabajo.
- Configuración que proporciona como parte de la suya `runtimeConfiguration` al crear una aplicación.
- Configuraciones optimizadas que Amazon EMR utiliza para la versión.
- Configuraciones de código abierto predeterminadas para la aplicación.

Para obtener más información sobre la declaración de configuraciones a nivel de aplicación y la anulación de las configuraciones durante la ejecución de un trabajo, consulteConfiguración de aplicaciones predeterminada para Serverless EMR.

Impulse la optimización dinámica de la asignación de recursos

Se utiliza `dynamicAllocationOptimization` para optimizar el uso de los recursos en EMR Serverless. Al establecer esta propiedad `true` en la clasificación de configuración de Spark, EMR Serverless debe optimizar la asignación de recursos de los ejecutores para alinear mejor la velocidad a la que Spark solicita y cancela ejecutores con la velocidad a la que EMR Serverless crea y libera trabajadores. De este modo, EMR Serverless reutiliza a los trabajadores de forma más óptima en todas las etapas, lo que reduce los costes al ejecutar tareas con varias etapas y, al mismo tiempo, mantener el mismo rendimiento.

Esta propiedad está disponible en todas las EMR versiones de Amazon.

El siguiente es un ejemplo de clasificación de configuraciones con`dynamicAllocationOptimization`.

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

Tenga en cuenta lo siguiente si utiliza la optimización de asignación dinámica:

- Esta optimización está disponible para los trabajos de Spark para los que has activado la asignación dinámica de recursos.
- Para lograr la mejor rentabilidad, te recomendamos configurar un límite de escalado superior para los trabajadores, utilizando la configuración a nivel de trabajo `spark.dynamicAllocation.maxExecutors` o la configuración de capacidad máxima a nivel de aplicación en función de tu carga de trabajo.
- Es posible que no vea una mejora de costes en los trabajos más sencillos. Por ejemplo, si tu trabajo se ejecuta en un conjunto de datos pequeño o termina de ejecutarse en una etapa, es posible que Spark no necesite un mayor número de ejecutores ni varios eventos de escalado.
- Los trabajos con una secuencia de una etapa grande, etapas más pequeñas y, después, una etapa más grande pueden experimentar una regresión en el tiempo de ejecución del trabajo. Dado que EMR Serverless utiliza los recursos de manera más eficiente, es posible que haya menos trabajadores disponibles para las etapas más grandes, lo que prolongará el tiempo de ejecución.

Aumente las propiedades laborales

La siguiente tabla muestra las propiedades opcionales de Spark y sus valores predeterminados que puedes anular al enviar un trabajo de Spark.

Clave	Descripción	Valor predeterminado
`spark.archives`	Una lista de archivos separados por comas que Spark extrae en el directorio de trabajo de cada ejecutor. Los tipos de archivos compatibles incluyen `.jar`, y `.tar.gz`. `.tgz` `.zip` Para especificar el nombre del directorio que se va a extraer, añádalo # después del nombre del archivo que desee extraer. Por ejemplo, `file.zip#directory` .	`NULL`

Clave	Descripción	Valor predeterminado
`spark.authenticate`	Opción que activa la autenticación de las conexiones internas de Spark.	`TRUE`
`spark.driver.cores`	El número de núcleos que utiliza el controlador.	4
`spark.driver.extraJavaOptions`	Opciones de Java adicionales para el controlador Spark.	`NULL`
`spark.driver.memory`	La cantidad de memoria que utiliza el controlador.	14 G
`spark.dynamicAllocation.enabled`	Opción que activa la asignación dinámica de recursos. Esta opción aumenta o reduce el número de ejecutores registrados en la aplicación, en función de la carga de trabajo.	`TRUE`
`spark.dynamicAllocation.executorIdleTimeout`	El tiempo que un ejecutor puede permanecer inactivo antes de que Spark lo elimine. Esto solo se aplica si activas la asignación dinámica.	60
`spark.dynamicAllocation.initialExecutors`	El número inicial de ejecutores que se ejecutarán si activas la asignación dinámica.	3

Clave	Descripción	Valor predeterminado
`spark.dynamicAllocation.maxExecutors`	El límite superior del número de ejecutores si activas la asignación dinámica.	Para la versión 6.10.0 y versiones posteriores, `infinity` Para la versión 6.9.0 y versiones anteriores, `100`
`spark.dynamicAllocation.minExecutors`	El límite inferior del número de ejecutores si se activa la asignación dinámica.	`0`
`spark.emr-serverless.allocation.batch.size`	El número de contenedores que se van a solicitar en cada ciclo de asignación de ejecutores. Hay un intervalo de un segundo entre cada ciclo de asignación.	20
`spark.emr-serverless.driver.disk`	El disco del controlador Spark.	20 G
`spark.emr-serverless.driverEnv.` *`[KEY]`*	Opción que añade variables de entorno al controlador Spark.	`NULL`
`spark.emr-serverless.executor.disk`	El disco ejecutor de Spark.	20 G
`spark.emr-serverless.memoryOverheadFactor`	Establece la sobrecarga de memoria para añadirla a la memoria del contenedor del controlador y del ejecutor.	0.1
`spark.emr-serverless.driver.disk.type`	El tipo de disco conectado al controlador Spark.	Estándar

Clave	Descripción	Valor predeterminado
`spark.emr-serverless.executor.disk.type`	El tipo de disco conectado a los ejecutores de Spark.	Estándar
`spark.executor.cores`	El número de núcleos que utiliza cada ejecutor.	4
`spark.executor.extraJavaOptions`	Opciones de Java adicionales para el ejecutor Spark.	NULL
`spark.executor.instances`	El número de contenedores ejecutores de Spark que se van a asignar.	3
`spark.executor.memory`	La cantidad de memoria que utiliza cada ejecutor.	14 G
`spark.executorEnv.`*`[KEY]`*	Opción que añade variables de entorno a los ejecutores de Spark.	NULL
`spark.files`	Una lista de archivos separados por comas para incluir en el directorio de trabajo de cada ejecutor. Puede acceder a las rutas de estos archivos en el ejecutor con. `SparkFiles.get(`*`fileName`*`)`	NULL
`spark.hadoop.hive.metastore.client.factory.class`	La clase de implementación de Hive Metastore.	NULL

Clave	Descripción	Valor predeterminado
`spark.jars`	Jarros adicionales para añadirlos a la ruta de clases en tiempo de ejecución del controlador y los ejecutores.	`NULL`
`spark.network.crypto.enabled`	Opción que activa el cifrado basado en datos. AES RPC Esto incluye el protocolo de autenticación agregado en Spark 2.2.0.	`FALSE`
`spark.sql.warehouse.dir`	La ubicación predeterminada para las bases de datos y tablas administradas.	El valor de `$PWD/spark-warehouse`
`spark.submit.pyFiles`	Una lista separada por comas de `.zip.egg`, o `.py` archivos para colocarlos en las aplicaciones de `PYTHONPATH` Python.	`NULL`

La siguiente tabla muestra los parámetros de envío predeterminados de Spark.

Clave	Descripción	Valor predeterminado
`archives`	Una lista de archivos separados por comas que Spark extrae en el directorio de trabajo de cada ejecutor.	`NULL`
`class`	La clase principal de la aplicación (para aplicaciones Java y Scala).	`NULL`

Clave	Descripción	Valor predeterminado
`conf`	Una propiedad de configuración arbitraria de Spark.	`NULL`
`driver-cores`	El número de núcleos que utiliza el controlador.	4
`driver-memory`	La cantidad de memoria que utiliza el controlador.	14 G
`executor-cores`	El número de núcleos que utiliza cada ejecutor.	4
`executor-memory`	La cantidad de memoria que utiliza el ejecutor.	14 G
`files`	Una lista de archivos separados por comas para colocarlos en el directorio de trabajo de cada ejecutor. Puede acceder a las rutas de estos archivos en el ejecutor con. `SparkFiles.get(` *`fileName`*`)`	`NULL`
`jars`	Lista de archivos jar separados por comas para incluirlos en las rutas de clases del controlador y del ejecutor.	`NULL`
`num-executors`	El número de ejecutores que se van a lanzar.	3

Clave	Descripción	Valor predeterminado
`py-files`	Una lista separada por comas de `.zip.egg`, o `.py` archivos para colocarlos en las aplicaciones de `PYTHONPATH` Python.	`NULL`
`verbose`	Opción que activa una salida de depuración adicional.	`NULL`

Ejemplos de Spark

El siguiente ejemplo muestra cómo utilizar el `StartJobRun` API para ejecutar un script de Python. Para ver un end-to-end tutorial que utiliza este ejemplo, consulteIntroducción a Amazon EMR Serverless. Puedes encontrar ejemplos adicionales de cómo ejecutar PySpark trabajos y añadir dependencias de Python en el repositorio 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"
        }
    }'
```

El siguiente ejemplo muestra cómo usarlo `StartJobRun` API para ejecutar un Spark. JAR

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
```

```
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
    }'
```

Empleos en Hive

Puede ejecutar trabajos de Hive en una aplicación con el `type` parámetro establecido en. `HIVE` Los trabajos deben ser compatibles con la versión de Hive compatible con la versión de EMR lanzamiento de Amazon. Por ejemplo, cuando ejecuta trabajos en una aplicación con la EMR versión 6.6.0 de Amazon, su trabajo debe ser compatible con Apache Hive 3.1.2. Para obtener información sobre las versiones de la aplicación para cada versión, consulte. Versiones de lanzamiento de Amazon EMR Serverless

Parámetros de trabajo de Hive

Cuando utilice el `StartJobRun`APIpara ejecutar un trabajo de Hive, debe especificar los siguientes parámetros.

Parámetros necesarios

- Función de ejecución de tareas de Hive
- Parámetro del controlador de tareas de Hive
- Parámetro de anulación de la configuración de Hive

Función de ejecución de tareas de Hive

`executionRoleArn`Utilícelo ARN para especificar el IAM rol que usa su aplicación para ejecutar los trabajos de Hive. Este rol debe contener los siguientes permisos:

- Lea desde los buckets de S3 u otras fuentes de datos en las que residen sus datos
- Lea los buckets o prefijos de S3 donde residen el archivo de consulta de Hive y el archivo de consulta de inicio
- Lea y escriba en los depósitos de S3 donde se encuentran el directorio de Hive Scratch y el directorio de almacén de Hive Metastore
- Escriba en los cubos S3 en los que desee escribir el resultado final

- Escribe los registros en un bucket o prefijo de S3 que especifique `S3MonitoringConfiguration`
- Acceda a KMS las claves si las usa KMS para cifrar los datos de su bucket de S3
- Acceso al AWS Catálogo de datos de Glue

Si su trabajo de Hive lee o escribe datos en o desde otras fuentes de datos, especifique los permisos adecuados en esta IAM función. Si no proporciona estos permisos al IAM rol, es posible que su trabajo no funcione. Para obtener más información, consulte Funciones de tiempo de ejecución de trabajos para Amazon EMR Serverless.

Parámetro del controlador de tareas de Hive

Se utiliza **`jobDriver`**para proporcionar información al trabajo. El parámetro del controlador de tareas solo acepta un valor para el tipo de trabajo que desee ejecutar. `hive`Al especificar el tipo de trabajo, EMR Serverless pasa una consulta de Hive al `jobDriver` parámetro. Los trabajos de Hive tienen los siguientes parámetros:

- **`query`**— Esta es la referencia en Amazon S3 al archivo de consulta de Hive que desea ejecutar.
- **`parameters`**— Estas son las propiedades de configuración adicionales de Hive que desea anular. Para anular las propiedades, páselas a este parámetro como. `--hiveconf` *`property=value`* Para anular las variables, páselas a este parámetro como. `--hivevar` *`key=value`*
- **`initQueryFile`**— Este es el archivo de consulta de init Hive. Hive ejecuta este archivo antes de la consulta y puede usarlo para inicializar tablas.

Parámetro de anulación de la configuración de Hive

Se utiliza **`configurationOverrides`**para anular las propiedades de configuración a nivel de supervisión y de aplicación. Este parámetro acepta un JSON objeto con los dos campos siguientes:

- **`monitoringConfiguration`**— Utilice este campo para especificar el Amazon S3 URL (`s3MonitoringConfiguration`) en el que desea que el trabajo EMR sin servidor almacene los registros de su trabajo de Hive. Asegúrese de crear este depósito con el mismo Cuenta de AWS que aloja su aplicación, y en el mismo Región de AWS donde se ejecuta su trabajo.
- **`applicationConfiguration`**— Puede proporcionar un objeto de configuración en este campo para anular las configuraciones predeterminadas de las aplicaciones. Puede utilizar una sintaxis abreviada para proporcionar la configuración o puede hacer referencia al objeto de configuración

en un archivo. JSON Los objetos de configuración se componen de una clasificación, propiedades y configuraciones anidadas opcionales. Las propiedades consisten en las configuraciones que desea anular en ese archivo. Puede especificar varias clasificaciones para varias aplicaciones en un único objeto. JSON

> **Note**
>
> Las clasificaciones de configuración disponibles varían según la versión específica de EMR Serverless. Por ejemplo, las clasificaciones para el Log4j personalizado solo `spark-executor-log4j2` están disponibles en las versiones 6.8.0 `spark-driver-log4j2` y superiores.

Si se pasa la misma configuración en la anulación de una aplicación y en los parámetros de Hive, los parámetros de Hive tienen prioridad. La siguiente lista clasifica las configuraciones de mayor a menor prioridad.

- Configuración que se proporciona como parte de los parámetros de Hive. `--hiveconf property=value`
- La configuración que se proporciona como parte de la aplicación se anula al iniciar un trabajo.
- Configuración que proporciona como parte de la suya `runtimeConfiguration` al crear una aplicación.
- Configuraciones optimizadas que Amazon EMR asigna para la versión.
- Configuraciones de código abierto predeterminadas para la aplicación.

Para obtener más información sobre cómo declarar las configuraciones a nivel de la aplicación y anular las configuraciones durante la ejecución de un trabajo, consulte. Configuración de aplicaciones predeterminada para Serverless EMR

Hive las propiedades de los trabajos

En la siguiente tabla se enumeran las propiedades obligatorias que debe configurar al enviar un trabajo de Hive.

Opción	Descripción
`hive.exec.scratchdir`	La ubicación de Amazon S3 en la que EMR Serverless crea archivos temporales durante la ejecución del trabajo de Hive.
`hive.metastore.warehouse.dir`	La ubicación de Amazon S3 de las bases de datos para las tablas administradas en Hive.

La siguiente tabla muestra las propiedades opcionales de Hive y sus valores predeterminados que puede anular al enviar un trabajo de Hive.

Opción	Descripción	Valor predeterminado
`fs.s3.customAWSCredentialsProvider`	La AWS El proveedor de credenciales que desea utilizar.	com.amazonaws.auth.d efaultAWSCredentialsProviderChain
`fs.s3a.aws.credentials.provider`	La AWS Proveedor de credenciales que desea utilizar con un sistema de archivos S3A.	com.amazonAWS.auth.D efaultAWSCredentialsProviderChain
`hive.auto.convert.join`	Opción que activa la conversión automática de las uniones comunes en uniones de mapa, en función del tamaño del archivo de entrada.	`TRUE`
`hive.auto.convert.join.noconditionaltask`	Opción que activa la optimización cuando Hive convierte una unión común en una unión de mapas en función del tamaño del archivo de entrada.	`TRUE`

Opción	Descripción	Valor predeterminado
`hive.auto.convert.join.noconditionaltask.size`	Una unión se convierte directamente en una unión de mapa de un tamaño inferior a este tamaño.	El valor óptimo se calcula en función de la memoria de tareas de Tez
`hive.cbo.enable`	Opción que activa las optimizaciones basadas en los costes con el marco Calcite.	`TRUE`
`hive.cli.tez.session.async`	Opción para iniciar una sesión de Tez en segundo plano mientras se compila la consulta de Hive. Si se configura en`false`, Tez AM se inicia después de compilar la consulta de Hive.	`TRUE`
`hive.compute.query.using.stats`	Opción que activa a Hive para responder a determinadas consultas con estadísticas almacenadas en el metabastore. Para obtener estadísticas básicas, configúrelo en`hive.stats.autogather` . TRUE Para obtener una colección de consultas más avanzada, ejecute`analyze table queries`.	`TRUE`

Opción	Descripción	Valor predeterminado
`hive.default.fileformat`	El formato de archivo predeterminado para `CREATE TABLE` las declaraciones. Puede anular esto de forma explícita si lo especifica `STORED AS [FORMAT]` en su `CREATE TABLE` comando.	`TEXTFILE`
`hive.driver.cores`	El número de núcleos que se van a utilizar para el proceso del controlador Hive.	2
`hive.driver.disk`	El tamaño del disco del controlador Hive.	20 G
`hive.driver.disk.type`	El tipo de disco del controlador Hive.	Estándar
`hive.tez.disk.type`	El tamaño del disco de los trabajadores del té.	Estándar
`hive.driver.memory`	La cantidad de memoria que se va a utilizar por proceso del controlador Hive. Hive CLI y Tez Application Master comparten esta memoria a partes iguales con un 20% de espacio libre.	6 G
`hive.emr-serverless.launch.env.[ KEY]`	Opción para configurar la variable de *KEY* entorno en todos los procesos específicos de Hive, como el controlador de Hive, Tez AM y la tarea de Tez.	

Opción	Descripción	Valor predeterminado
`hive.exec.dynamic.partition`	Opciones que activan las particiones dinámicas en/. DML DDL	`TRUE`
`hive.exec.dynamic.partition.mode`	Opción que especifica si desea utilizar el modo estricto o el modo no estricto. En el modo estricto, debe especific ar al menos una partición estática en caso de que sobrescriba accidentalmente todas las particiones. En el modo no estricto, todas las particiones pueden ser dinámicas.	`strict`
`hive.exec.max.dynamic.partitions`	El número máximo de particiones dinámicas que Hive crea en total.	1 000
`hive.exec.max.dynamic.partitions.pernode`	Número máximo de particion es dinámicas que Hive crea en cada nodo mapeador y reductor.	100

Opción	Descripción	Valor predeterminado
`hive.exec.orc.split.strategy`	Espera uno de los siguientes valores:`BI`,, `ETL` o. `HYBRID` No se trata de una configuración a nivel de usuario. `BI`especifica que desea dedicar menos tiempo a la generación dividida en lugar de a la ejecución de consultas. `ETL`especifica que desea dedicar más tiempo a la generación dividida. `HYBRID`Especifica una selección de las estrategias anteriores en función de la heurística.	`HYBRID`
`hive.exec.reducers.bytes.per.reducer`	El tamaño por reductor. El valor predeterminado es 256 MB. Si el tamaño de entrada es de 1 G, el trabajo utiliza 4 reductores.	256000000
`hive.exec.reducers.max`	El número máximo de reductores.	256
`hive.exec.stagingdir`	El nombre del directorio que almacena los archivos temporales que Hive crea dentro de las ubicaciones de las tablas y en la ubicación del directorio temporal especificada en la `hive.exec.scratchdir` propiedad.	`.hive-staging`

Opción	Descripción	Valor predeterminado
`hive.fetch.task.conversion`	Espera uno de los siguientes valores:`NONE`,`MINIMAL`, o`MORE`. Hive puede convertir consultas seleccionadas en una sola `FETCH` tarea. Esto minimiza la latencia.	`MORE`
`hive.groupby.position.alias`	Opción que hace que Hive utilice un alias de posición de columna en `GROUP BY` las declaraciones.	`FALSE`
`hive.input.format`	El formato de entrada predeterminado. `HiveInputFormat` Configúrelo en si tiene problemas con`CombineHiveInputFormat` .	`org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`
`hive.log.explain.output`	Opción que activa las explicaciones de los resultados ampliados para cualquier consulta del registro de Hive.	`FALSE`
`hive.log.level`	El nivel de registro de Hive.	`INFO`
`hive.mapred.reduce.tasks.speculative.execution`	Opción que activa el lanzamiento especulativo de los reductores. Solo es compatible con Amazon EMR 6.10.x y versiones anteriores.	`TRUE`

Opción	Descripción	Valor predeterminado
`hive.max-task-containers`	El número máximo de contenedores simultáneos. La memoria del mapeador configurada se multiplica por este valor para determinar la memoria disponible que divide el uso de cómputo y de prioridad de tareas.	1 000
`hive.merge.mapfiles`	Opción que hace que los archivos pequeños se fusionen al final de un trabajo solo de mapas.	`TRUE`
`hive.merge.size.per.task`	El tamaño de los archivos fusionados al final del trabajo.	256000000
`hive.merge.tezfiles`	Opción que activa la combinación de archivos pequeños al final de un TezDAG.	`FALSE`
`hive.metastore.client.factory.class`	El nombre de la clase de fábrica que produce los objetos que implementan la `IMetaStoreClient` interfaz.	`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`
`hive.metastore.glue.catalogid`	Si el archivo de AWS Glue Data Catalog actúa como un metaalmacén, pero se ejecuta en un sitio diferente Cuenta de AWS a diferencia de los trabajos, el ID del Cuenta de AWS dónde se ejecutan los trabajos.	`NULL`

Opción	Descripción	Valor predeterminado
`hive.metastore.uris`	El ahorro URI que utiliza el cliente de Metastore para conectarse a un Metastore remoto.	`NULL`
`hive.optimize.ppd`	Opción que activa la pulsación de predicados.	`TRUE`
`hive.optimize.ppd.storage`	Opción que activa la inserción de predicados en los controladores de almacenamiento.	`TRUE`
`hive.orderby.position.alias`	Opción que hace que Hive utilice un alias de posición de columna en las declaraciones. `ORDER BY`	`TRUE`
`hive.prewarm.enabled`	Opción que activa el precalentamiento del recipiente para Tez.	`FALSE`
`hive.prewarm.numcontainers`	El número de recipientes que se deben precalentar para el té.	10
`hive.stats.autogather`	Opción que hace que Hive recopile estadísticas básicas automáticamente durante el `INSERT OVERWRITE` comando.	`TRUE`

Opción	Descripción	Valor predeterminado
`hive.stats.fetch.column.stats`	Opción que desactiva la búsqueda de estadísticas de columnas desde el metabastore. La búsqueda de estadísticas de columnas puede resultar costosa cuando el número de columnas es elevado.	`FALSE`
`hive.stats.gather.num.threads`	El número de subprocesos que utilizan los `partialscan` comandos `noscan` Analyze para las tablas particionadas. Esto solo se aplica a los formatos de archivo que implementan `StatsProvidingRecordReader` (comoORC).	10
`hive.strict.checks.cartesian.product`	Opciones que activan las comprobaciones de unión cartesianas estrictas. Estas comprobaciones no permiten un producto cartesiano (una unión cruzada).	`FALSE`
`hive.strict.checks.type.safety`	Opción que activa los controles de seguridad de tipo estricto y desactiva la `bigint` comparación entre y. `string` `double`	`TRUE`

Opción	Descripción	Valor predeterminado
`hive.support.quoted.identifiers`	Espera un valor de `NONE` o`COLUMN`. `NONE`implica que solo los caracteres alfanuméricos y de subrayado son válidos en los identificadores. `COLUMN`implica que los nombres de las columnas pueden contener cualquier carácter.	`COLUMN`
`hive.tez.auto.reducer.parallelism`	Opción que activa la función de paralelismo del autorreductor Tez. Hive sigue estimando los tamaños de los datos y establece estimaciones de paralelismo. Tez toma muestras de los tamaños de salida de los vértices de la fuente y ajusta las estimaciones en tiempo de ejecución según sea necesario.	`TRUE`
`hive.tez.container.size`	La cantidad de memoria que se va a utilizar por proceso de tareas de Tez.	6144
`hive.tez.cpu.vcores`	El número de núcleos que se van a utilizar para cada tarea de Tez.	2
`hive.tez.disk.size`	El tamaño del disco de cada contenedor de tareas.	20 G
`hive.tez.input.format`	El formato de entrada para la generación de divisiones en el Tez AM.	`org.apache.hadoop.hive.ql.io.HiveInputFormat`

Opción	Descripción	Valor predeterminado
`hive.tez.min.partition.factor`	Límite inferior de reductores que Tez especifica al activar el paralelismo del autorreductor.	0,25
`hive.vectorized.execution.enabled`	Opción que activa el modo vectorizado de ejecución de consultas.	`TRUE`
`hive.vectorized.execution.reduce.enabled`	Opción que activa el modo vectorizado del lado reducido de la ejecución de una consulta.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	El nombre de la clase de controlador de un metaalmacén. JDBC	`org.apache.derby.jdbc.EmbeddedDriver`
`javax.jdo.option.ConnectionPassword`	La contraseña asociada a una base de datos de metastore.	`NULL`
`javax.jdo.option.ConnectionURL`	La cadena de JDBC conexión de un JDBC metaalmacén.	`jdbc:derby:;databaseName=metastore_db;create=true`
`javax.jdo.option.ConnectionUserName`	El nombre de usuario asociado a una base de datos de metastore.	`NULL`
`mapreduce.input.fileinputformat.split.maxsize`	El tamaño máximo de una división durante el cálculo dividido cuando el formato de entrada es. `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat` Un valor de 0 indica que no hay ningún límite.	0

Opción	Descripción	Valor predeterminado
`tez.am.dag.cleanup.on.completion`	Opción que activa la limpieza de los datos aleatorios cuando se completa. DAG	`TRUE`
`tez.am.emr-serverless.launch.env.[KEY]`	Opción para configurar la variable de *KEY* entorno en el proceso de Tez AM. En el caso de Tez AM, este valor anula el `hive.emr-serverless.launch.env.[KEY]` valor.	
`tez.am.log.level`	El nivel de registro raíz que EMR Serverless pasa al maestro de aplicaciones de Tez.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMRServerless debería enviar ATS los eventos después de este período de tiempo tras una solicitud de cierre de AM.	0
`tez.am.speculation.enabled`	Opción que provoca el lanzamiento especulativo de tareas más lentas. Esto puede ayudar a reducir la latencia de las tareas cuando algunas tareas se ejecutan más lentamente debido a máquinas defectuosas o lentas. Solo es compatible con Amazon EMR 6.10.x y versiones anteriores.	`FALSE`

Opción	Descripción	Valor predeterminado
`tez.am.task.max.failed.attempts`	El número máximo de intentos que pueden fallar para una tarea en particular antes de que la tarea falle. Este número no cuenta los intentos finalizados manualmente.	3
`tez.am.vertex.cleanup.height`	Distancia a la que, si todos los vértices dependientes están completos, Tez AM eliminará los datos de mezcla de vértices. Esta función se desactiva cuando el valor es 0. EMRLas versiones 6.8.0 y posteriores de Amazon admiten esta función.	0
`tez.client.asynchronous-stop`	Opción que hace que EMR Serverless envíe ATS eventos antes de que finalice el controlador Hive.	`FALSE`
`tez.grouping.max-size`	El límite de tamaño superior (en bytes) de una división agrupada. Este límite evita que las divisiones sean excesivamente grandes.	1073741824
`tez.grouping.min-size`	El límite de tamaño inferior (en bytes) de una división agrupada. Este límite evita que se produzcan demasiadas divisiones pequeñas.	16777216

Opción	Descripción	Valor predeterminado
`tez.runtime.io.sort.mb`	El tamaño del búfer flexible cuando Tez ordena la salida se ordena.	El valor óptimo se calcula en función de la memoria de tareas de Tez
`tez.runtime.unordered.output.buffer.size-mb`	El tamaño del búfer que se utilizará si Tez no escribe directamente en el disco.	El valor óptimo se calcula en función de la memoria de tareas de Tez
`tez.shuffle-vertex-manager.max-src-fraction`	La fracción de tareas de origen que debe completarse antes de que EMR Serverless programe todas las tareas para el vértice actual (en el caso de una `ScatterGather` conexión). La cantidad de tareas listas para su programación en el vértice actual se escala linealmente entre y. `min-fraction` `max-fraction` Este valor predeterminado es el valor predeterminado o el que sea `tez.shuffle-vertex-manager.min-src-fraction` mayor.	0.75
`tez.shuffle-vertex-manager.min-src-fraction`	La fracción de tareas de origen que deben completarse antes de que EMR Serverless programe las tareas para el vértice actual (en caso de una `ScatterGather` conexión).	0,25

Opción	Descripción	Valor predeterminado
`tez.task.emr-serverless.launch.env.[ KEY]`	Opción para configurar la variable de *KEY* entorno en el proceso de tareas de Tez. Para las tareas de Tez, este valor anula el `hive.emr-serverless.launch.env.[ KEY]` valor.	
`tez.task.log.level`	El nivel de registro raíz que EMR Serverless transfiere a las tareas de Tez.	`INFO`
`tez.yarn.ats.event.flush.timeout.millis`	El tiempo máximo que AM debe esperar a que se eliminen los eventos antes de apagarse.	300.000

Ejemplos de trabajos de Hive

El siguiente ejemplo de código muestra cómo ejecutar una consulta de Hive con. `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"
            }
        }]
    }'
```

Puede encontrar ejemplos adicionales de cómo ejecutar trabajos de Hive en el repositorio EMRServerless Samples. GitHub

EMRResiliencia laboral sin servidor

EMRLas versiones 7.1.0 y posteriores de Serverless incluyen compatibilidad con la resiliencia de los trabajos, por lo que reintenta automáticamente cualquier trabajo fallido sin necesidad de intervención manual por tu parte. Otra ventaja de la resiliencia laboral es que EMR Serverless traslada las ejecuciones de tareas a una zona de disponibilidad (AZ) diferente en caso de que una zona de disponibilidad experimente algún problema.

Para habilitar la resiliencia laboral de un trabajo, establezca la política de reintentos para su trabajo. Una política de reintentos garantiza que EMR Serverless reinicie automáticamente un trabajo en caso de que se produzca un error en algún momento. Las políticas de reintento se admiten tanto para los trabajos por lotes como para los de streaming, por lo que puede personalizar la resiliencia de los trabajos en función de su caso de uso. En la siguiente tabla se comparan los comportamientos y las diferencias en cuanto a la resiliencia de los trabajos en lotes y en streaming.

	Tareas por lotes	Transmitiendo trabajos
Comportamiento predeterminado	No vuelve a ejecutar el trabajo.	Siempre vuelve a intentar ejecutar el trabajo, ya que la aplicación crea puntos de control mientras se ejecuta el trabajo.
Punto de reintento	Los trabajos por lotes no tienen puntos de control, por lo que EMR Serverless	Los trabajos de streaming admiten puntos de control, por lo que puede configurar

	Tareas por lotes	Transmitiendo trabajos
	siempre vuelve a ejecutar el trabajo desde el principio.	la consulta de streaming para guardar el estado del tiempo de ejecución y el progreso hasta una ubicación de punto de control en Amazon S3. EMRServerless reanuda la ejecución del trabajo desde el punto de control. Para obtener más información, consulte Cómo recuperarse de errores con Checkpoints en la documentación de Apache Spark.
Número máximo de intentos de reintento	Permite un máximo de 10 reintentos.	Los trabajos de streaming tienen un control de prevención de golpes integrado, por lo que la aplicación deja de volver a intentar los trabajos si siguen fallando después de una hora. El número predeterminado de reintentos en una hora es de cinco intentos. Puede configurar este número de reintentos para que esté comprendido entre 1 y 10. No puedes personalizar el número máximo de intentos. Un valor de 1 indica que no hay reintentos.

Cuando EMR Serverless intenta volver a ejecutar un trabajo, también indexa el trabajo con un número de intentos, para que pueda realizar un seguimiento del ciclo de vida de un trabajo en todos sus intentos.

Puede utilizar las operaciones EMR sin servidor API o las AWS CLI para cambiar la resiliencia laboral o ver información relacionada con la resiliencia laboral. Para obtener más información, consulte la guía EMRServerless API.

De forma predeterminada, EMR Serverless no vuelve a ejecutar los trabajos por lotes. Para habilitar los reintentos de los trabajos por lotes, configure el `maxAttempts` parámetro al iniciar la ejecución de un trabajo por lotes. El `maxAttempts` parámetro solo se aplica a los trabajos por lotes. El valor predeterminado es 1, lo que significa que no se debe volver a ejecutar el trabajo. Los valores aceptados son del 1 al 10, ambos inclusive.

El siguiente ejemplo muestra cómo especificar un número máximo de 10 intentos al iniciar la ejecución de un trabajo.

```
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"
     }
}'
```

EMRServerless vuelve a intentar transmitir los trabajos de streaming de forma indefinida si fallan. Para evitar que se produzcan errores repetidos e irrecuperables, utilice el control de prevención de atascos `maxFailedAttemptsPerHour` para los reintentos de tareas de streaming. Este parámetro le permite especificar el número máximo de intentos fallidos permitidos una hora antes de que Serverless deje de reintentarlo. EMR El valor predeterminado es cinco. Los valores aceptados son del 1 al 10, ambos 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"
     }
}'
```

También puede utilizar las demás API operaciones de ejecución de tareas para obtener información sobre las tareas. Por ejemplo, puede usar el `attempt` parámetro con la `GetJobRun` operación para obtener detalles sobre un intento de trabajo específico. Si no incluye el `attempt` parámetro, la operación devuelve información sobre el último intento.

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

La `ListJobRunAttempts` operación devuelve información sobre todos los intentos relacionados con la ejecución de un trabajo.

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

La `GetDashboardForJobRun` operación crea y devuelve un URL objeto que puede utilizar para acceder a la aplicación UIs y ejecutar un trabajo. El `attempt` parámetro le permite obtener un URL para un intento específico. Si no incluye el `attempt` parámetro, la operación devuelve información sobre el último intento.

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

```
    --attempt 1
```

Supervisión de un trabajo con una política de reintento

La compatibilidad con la resiliencia de los trabajos también añade el nuevo evento EMRServerless Job Run Retry. EMRServerless publica este evento en cada reintento del trabajo. Puede usar esta notificación para realizar un seguimiento de los reintentos del trabajo. Para obtener más información sobre los eventos, consulta Amazon EventBridge events.

Política de registro con reintentos

Cada vez que EMR Serverless vuelve a intentar un trabajo, el intento genera su propio conjunto de registros. Para garantizar que EMR Serverless pueda entregar correctamente estos registros a Amazon S3 y Amazon CloudWatch sin sobrescribir ninguno, EMR Serverless añade un prefijo al formato de la ruta de registro de S3 y al nombre del flujo de CloudWatch registro para incluir el número de intento del trabajo.

A continuación se muestra un ejemplo del aspecto del formato.

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

Este formato garantiza que EMR Serverless publique todos los registros de cada intento de trabajo en su propia ubicación designada en Amazon S3 y CloudWatch. Para obtener más información, consulte Almacenamiento de registros.

> **Note**
>
> EMRServerless solo usa este formato de prefijo en todos los trabajos de streaming y en los trabajos por lotes que tengan habilitada la opción de reintento.

Configuración de Metastore

Un metaalmacén de Hive es una ubicación centralizada que almacena información estructural sobre las tablas, incluidos los esquemas, los nombres de las particiones y los tipos de datos. Con EMR Serverless, puedes conservar los metadatos de esta tabla en un metaalmacén que tenga acceso a tus trabajos.

Tienes dos opciones para crear un metaalmacén de Hive:

- La AWS Catálogo de datos de Glue
- Un metaalmacén externo de Apache Hive

Uso de AWS Glue Data Catalog como metastore

Puedes configurar tus trabajos de Spark y Hive para usar el AWS Glue Data Catalog es su metatienda. Recomendamos esta configuración cuando necesite un metaalmacén persistente o compartido por diferentes aplicaciones, servicios o Cuentas de AWS. Para obtener más información sobre el catálogo de datos, consulte Rellenar el AWS Catálogo de datos de Glue. Para obtener más información AWS Precios de Glue, consulte AWS Precios de Glue.

Puede configurar su trabajo EMR sin servidor para usar el AWS Glue Data Catalog en el mismo Cuenta de AWS como su aplicación o en una diferente Cuenta de AWS.

Configure el AWS Catálogo de datos de Glue

Para configurar el catálogo de datos, elija el tipo de aplicación EMR sin servidor que desee utilizar.

Spark

Cuando usas EMR Studio para ejecutar tus trabajos con aplicaciones Spark EMR sin servidor, el AWS Glue Data Catalog es el metabastore predeterminado.

Cuando usas SDKs o AWS CLI, puede establecer la `spark.hadoop.hive.metastore.client.factory.class` configuración `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` en los `sparkSubmit` parámetros de la ejecución de su trabajo. El siguiente ejemplo muestra cómo configurar el catálogo de datos con 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, puedes establecer esta configuración al crear una nueva `SparkSession` en tu código de 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

En el EMR caso de las aplicaciones Hive sin servidor, el catálogo de datos es el metabastore predeterminado. Es decir, cuando se ejecutan trabajos en una aplicación Hive EMR sin servidor, Hive registra la información del metaalmacén del catálogo de datos en la misma Cuenta de AWS como su aplicación. No necesita una nube privada virtual (VPC) para usar el catálogo de datos como metaalmacén.

Para acceder a las tablas del metaalmacén de Hive, añada las tablas necesarias AWS Las políticas de Glue se describen en Configuración de IAM permisos para AWS Glue.

Configure el acceso multicuenta para EMR Serverless y AWS Catálogo de datos de Glue

Para configurar el acceso multicuenta para EMR Serverless, primero debe iniciar sesión en lo siguiente Cuentas de AWS:

- `AccountA`— Un Cuenta de AWS donde ha creado una aplicación EMR sin servidor.

- `AccountB`— Un Cuenta de AWS que contiene un AWS Glue Data Catalog al que desea que acceda su trabajo EMR sin servidor.

1. Asegúrese de que un administrador u otra identidad autorizada `AccountB` adjunte una política de recursos al catálogo de datos en. `AccountB` Esta política otorga permisos `AccountA` específicos entre cuentas para realizar operaciones con los recursos del `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. Agregue una IAM política a la función EMR Serverless Job Runtime `AccountA` para que esa función pueda acceder a los recursos del catálogo de datos. `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. Inicie la ejecución de su trabajo. Este paso es ligeramente diferente según el tipo `AccountA` de aplicación EMR sin servidor.

   Spark

   Defina la `spark.hadoop.hive.metastore.glue.catalogid` propiedad en la `hive-site` clasificación, tal y como se muestra en el siguiente ejemplo. Reemplazar *ID de catálogo de la cuenta* con el ID del catálogo de datos en. `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 la `hive.metastore.glue.catalogid` propiedad en la `hive-site` clasificación, tal y como se muestra en el siguiente ejemplo. Reemplazar *ID de catálogo de la cuenta* con el ID del catálogo de datos en. `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"
        }
    }]
}'
```

Consideraciones a la hora de utilizar el AWS Catálogo de datos de Glue

Puede añadir un elemento auxiliar JARs `ADD JAR` en sus scripts de Hive. Para obtener información adicional, consulte Consideraciones a la hora de utilizar AWS Catálogo de datos de Glue.

Uso de un metaalmacén de Hive externo

Puede configurar sus trabajos de Spark y Hive EMR sin servidor para que se conecten a un metaalmacén de Hive externo, como Amazon Aurora o Amazon for My. RDS SQL En esta sección,

se describe cómo configurar un metaalmacén de Amazon RDS Hive, configurar sus VPC trabajos EMR sin servidor y configurarlos para que usen un metaalmacén externo.

Cree un metastore de Hive externo

1. Cree una Amazon Virtual Private Cloud (AmazonVPC) con subredes privadas siguiendo las instrucciones de Create a VPC.
2. Cree su aplicación EMR sin servidor con sus nuevas subredes de Amazon VPC y privadas. Cuando configura su aplicación EMR sin servidor con unVPC, primero aprovisiona una interfaz de red elástica para cada subred que especifique. A continuación, conecta el grupo de seguridad especificado a esa interfaz de red. Esto le da a la aplicación el control de acceso. Para obtener más información sobre cómo configurar suVPC, consulteConfigurar VPC el acceso.
3. Cree una SQL base de datos My SQL o Aurora Postgre en una subred privada de Amazon. VPC Para obtener información sobre cómo crear una RDS base de datos de Amazon, consulte Creación de una RDS instancia de base de datos de Amazon.
4. Modifique el grupo de seguridad de su base de datos My SQL o Aurora para permitir JDBC las conexiones desde su grupo de seguridad EMR sin servidor siguiendo los pasos que se indican en Modificación de una RDS instancia de base de datos de Amazon. Añada una regla para el tráfico entrante al grupo de RDS seguridad desde uno de sus grupos de seguridad EMR sin servidor.

Tipo	Protocolo	Intervalo de puertos	Origen
Todos TCP	TCP	3306	`emr-serverless-security-group`

Configura las opciones de Spark

Usando JDBC

Para configurar su aplicación Spark EMR sin servidor para que se conecte a un metaalmacén de Hive basado en una SQL instancia Amazon RDS for My o Amazon SQL Aurora My, utilice una conexión. JDBC Introduce `mariadb-connector-java.jar` los `spark-submit` parámetros de `--jars` la ejecución de tu trabajo.

```
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/"
        }
    }
}'
```

El siguiente ejemplo de código es un script de punto de entrada de Spark que interactúa con una metatienda de Hive en 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()
```

¿Utilizas el servicio de segunda mano

Puede configurar su aplicación Hive EMR sin servidor para que se conecte a un metaalmacén de Hive basado en una instancia Amazon RDS for My o Amazon SQL Aurora My. SQL Para ello, ejecute un servidor de segunda mano en el nodo principal de un EMR clúster de Amazon existente. Esta opción es ideal si ya tiene un EMR clúster de Amazon con un servidor de segunda mano que desea utilizar para simplificar las configuraciones de sus trabajos EMR sin 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/"
        }
    }
}'
```

El siguiente ejemplo de código es un script de punto de entrada (`thriftscript.py`) que usa el protocolo thrift para conectarse a un metaalmacén de Hive. Tenga en cuenta que la `hive.metastore.uris` propiedad debe configurarse para que se lea desde un metaalmacén de Hive externo.

```
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()
```

Configure las opciones de Hive

Usando JDBC

Si desea especificar una ubicación de base de datos de Hive externa en una instancia de Amazon RDS My SQL o Amazon Aurora, puede anular la configuración predeterminada del metaalmacén.

> **Note**
>
> En Hive, puede realizar varias escrituras en tablas de metaalmacenes al mismo tiempo. Si compartes la información del metaalmacén entre dos trabajos, asegúrate de no escribir en la misma tabla de metastore simultáneamente, a menos que escribas en particiones diferentes de la misma tabla de metastore.

Defina las siguientes configuraciones en la `hive-site` clasificación para activar el metaalmacén externo de 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"
    }
}
```

Uso de un servidor de segunda mano

Puede configurar su aplicación Hive EMR sin servidor para que se conecte a un metaalmacén de Hive basado en Amazon RDS for My o Amazon SQL Aurora M. ySQLinstance Para ello, ejecuta un servidor de segunda mano en el nodo principal de un EMR clúster de Amazon existente. Esta opción es ideal si ya tienes un EMR clúster de Amazon que ejecuta un servidor de segunda mano y quieres usar tus configuraciones de trabajo EMR sin servidor.

Defina las siguientes configuraciones en la `hive-site` clasificación para que EMR Serverless pueda acceder al metaalmacén remoto de Thrift. Tenga en cuenta que debe configurar la `hive.metastore.uris` propiedad para que se lea desde un metaalmacén de Hive externo.

```
{
    "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"
    }
}
```

Consideraciones a la hora de utilizar un metaalmacén externo

- Puede configurar bases de datos que sean compatibles con MariaDB como su JDBC metaalmacén. Algunos ejemplos de estas bases RDS de datos son MariaDB, SQL My y Amazon Aurora.
- Los metastores no se inicializan automáticamente. Si tu metaalmacén no está inicializado con un esquema para tu versión de Hive, usa la herramienta de esquemas de Hive.
- EMRServerless no admite la autenticación Kerberos. No puedes usar un servidor metastore de segunda mano con la autenticación Kerberos con trabajos de Spark o Hive EMR sin servidor.

Acceder a los datos de S3 en otro AWS cuenta de EMR Serverless

Puede ejecutar trabajos de Amazon EMR Serverless desde una AWS crear una cuenta y configurarlos para acceder a los datos de los buckets de Amazon S3 que pertenecen a otro AWS account. En esta página, se describe cómo configurar el acceso multicuenta a S3 desde EMR Serverless.

Los trabajos que se ejecutan en EMR Serverless pueden usar una política de bucket de S3 o un rol asumido para acceder a los datos de Amazon S3 desde un sitio diferente. AWS account.

Requisitos previos

Para configurar el acceso multicuenta a Amazon EMR Serverless, debe completar las tareas con la sesión iniciada en dos AWS cuentas:

- **`AccountA`**— Este es el AWS cuenta en la que ha creado una aplicación Amazon EMR Serverless. Antes de configurar el acceso entre cuentas, debe tener lo siguiente preparado en esta cuenta:
  - Una aplicación de Amazon EMR Serverless en la que desee ejecutar trabajos.
  - Un rol de ejecución de trabajos que tiene los permisos necesarios para ejecutar trabajos en la aplicación. Para obtener más información, consulte Funciones de tiempo de ejecución de trabajos para Amazon EMR Serverless.
- **`AccountB`**— Este es el AWS cuenta que contiene el bucket de S3 al que desea que accedan sus trabajos de Amazon EMR Serverless.

Utilice una política de bucket de S3 para acceder a los datos de S3 entre cuentas

Para acceder al depósito de S3 en account B desde account A, adjunte la siguiente política al depósito de S3 en 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 obtener más información sobre el acceso entre cuentas de S3 con las políticas de bucket de S3, consulte el Ejemplo 2: El propietario del bucket concede permisos de bucket entre cuentas en la Guía del usuario de Amazon Simple Storage Service.

Utilice un rol asumido para acceder a los datos de S3 entre cuentas

Otra forma de configurar el acceso multicuenta a Amazon EMR Serverless es con la `AssumeRole` acción de AWS Security Token Service (AWS STS). AWS STS es un servicio web global que le permite solicitar credenciales temporales con privilegios limitados para los usuarios. Puede realizar API llamadas a EMR Serverless y Amazon S3 con las credenciales de seguridad temporales con `AssumeRole` las que las haya creado.

Los siguientes pasos ilustran cómo utilizar un rol asumido para acceder a los datos de S3 entre cuentas desde EMR Serverless:

1. Cree un bucket de Amazon S3, *`cross-account-bucket`*, en`AccountB`. Para obtener más información, consulte Creación de un bucket en la Guía del usuario de Amazon Simple Storage Service. Si desea tener acceso entre cuentas a DynamoDB, también puede crear una tabla de DynamoDB en la `AccountB`. Para obtener más información, consulte Creación de una tabla de DynamoDB en la Guía para desarrolladores de Amazon DynamoDB.
2. Cree un `Cross-Account-Role-B` IAM rol en el que pueda acceder a `AccountB` *`cross-account-bucket`*.
   a. Inicie sesión en AWS Management Console y abre la IAM consola en https://console.aws.amazon.com/iam/.
   b. Elija Roles y, a continuación, cree un nuevo rol: `Cross-Account-Role-B`. Para obtener más información sobre cómo crear IAM funciones, consulte Creación de IAM funciones en la Guía del IAM usuario.
   c. Cree una IAM política que especifique los permisos `Cross-Account-Role-B` para acceder al *`cross-account-bucket`* Un bucket de S3, como se muestra en la siguiente declaración de política. A continuación, adjunte la IAM política a`Cross-Account-Role-B`. Para obtener más información, consulte Creación de IAM políticas en la Guía del IAM usuario.

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

   Si necesita acceso a DynamoDB, cree IAM una política que especifique los permisos para acceder a la tabla de DynamoDB multicuenta. A continuación, adjunte la política a. IAM `Cross-Account-Role-B` Para obtener más información, consulte Amazon DynamoDB: permite el acceso a una tabla específica en IAM la Guía del usuario.

   La siguiente es una política para permitir el acceso a la tabla de DynamoDB`CrossAccountTable`.

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

3. Edite la relación de confianza del rol `Cross-Account-Role-B`.
   a. Para configurar la relación de confianza del rol, seleccione la pestaña Relaciones de confianza de la IAM consola para el rol `Cross-Account-Role-B` que creó en el paso 2.
   b. Seleccione Editar la relación de confianza.
   c. Agregue el siguiente documento de política. Esto le `AccountA` permite `Job-Execution-Role-A` asumir el `Cross-Account-Role-B` rol.

```
{
"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` en `AccountA` el AWS STS `AssumeRole`permiso para asumir`Cross-Account-Role-B`.
   a. En la IAM consola para AWS cuenta`AccountA`, selecciona`Job-Execution-Role-A`.
   b. Agregue la siguiente instrucción de política al `Job-Execution-Role-A` para denegar la acción `AssumeRole` en el rol `Cross-Account-Role-B`.

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

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

Ejemplos de roles asumidos

Puede usar un único rol asumido para acceder a todos los recursos de S3 de una cuenta o, con Amazon EMR 6.11 y versiones posteriores, puede configurar varios IAM roles para que los asuma al acceder a diferentes buckets de S3 entre cuentas.

Temas

- Acceda a los recursos de S3 asumiendo un rol
- Acceda a los recursos de S3 asumiendo varios roles

Acceda a los recursos de S3 asumiendo un rol

> **Note**
>
> Al configurar un trabajo para usar un único rol asumido, todos los recursos de S3 del trabajo usan ese rol, incluido el `entryPoint` script.

Si desea utilizar un único rol asumido para acceder a todos los recursos de S3 de la cuenta B, especifique las siguientes configuraciones:

1. Especifique EMRFS la configuración `fs.s3.customAWSCredentialsProvider` para`spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRol`
2. En el caso de Spark, utilice `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` y `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` especifique las variables de entorno del controlador y los ejecutores.

3. Para Hive `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, utilice y `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` especifique las variables de entorno en el controlador Hive, el maestro de aplicaciones de Tez y los contenedores de tareas de Tez.

Los siguientes ejemplos muestran cómo usar un rol asumido para iniciar la ejecución de un trabajo EMR sin servidor con acceso multicuenta.

Spark

En el siguiente ejemplo, se muestra cómo usar un rol asumido para iniciar la ejecución de un trabajo de EMR Serverless Spark con acceso multicuenta a 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

En el siguiente ejemplo, se muestra cómo usar un rol asumido para iniciar la ejecución de un trabajo de EMR Serverless Hive con acceso multicuenta a 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"
            }
        }]
    }'
```

Acceda a los recursos de S3 asumiendo varios roles

Con las versiones 6.11.0 y posteriores de EMR Serverless, puede configurar varios IAM roles para que los asuma al acceder a diferentes grupos de cuentas múltiples. Si desea acceder a diferentes recursos de S3 con diferentes funciones asumidas en la cuenta B, utilice las siguientes configuraciones al iniciar la ejecución del trabajo:

1. Especifique EMRFS la configuración `fs.s3.customAWSCredentialsProvider` para`com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCrede`

2. Especifique la EMRFS configuración `fs.s3.bucketLevelAssumeRoleMapping` para definir el mapeo entre el nombre del depósito de S3 y el IAM rol que se va a asumir en la cuenta B. El valor debe tener el formato de`bucket1->role1;bucket2->role2`.

Por ejemplo, puede usarlo `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` para acceder al bucket `bucket1` y usarlo `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` para acceder al bucket`bucket2`. Los siguientes ejemplos muestran cómo iniciar un trabajo EMR sin servidor ejecutado con acceso multicuenta a través de varios roles asumidos.

Spark

En el siguiente ejemplo, se muestra cómo utilizar varios roles asumidos para crear una ejecución de tareas de 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

Los siguientes ejemplos muestran cómo usar varios roles asumidos para crear una ejecución de tareas de 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"
            }
        }]
    }'
```

Solución de errores en EMR Serverless

Utilice la siguiente información para diagnosticar y solucionar problemas comunes que puedan surgir al trabajar con Amazon EMR Serverless.

Temas

- Error: se ha superado el límite de capacidad máxima permitida.
- Error: se ha superado la capacidad máxima configurada. Inténtelo de nuevo más tarde.
- Error: se ha denegado el acceso a S3. Compruebe los permisos de acceso a S3 del rol de ejecución del trabajo en los recursos de S3 necesarios.
- Error ModuleNotFoundError: No hay ningún módulo con nombre<module>. Consulte la guía del usuario sobre cómo utilizar las bibliotecas de Python con EMR Serverless.

- Error: no se pudo asumir la función de ejecución <role name>porque no existe o no está configurada con la relación de confianza requerida.

Error: se ha superado el límite de capacidad máxima permitida.

Este error indica que EMR Serverless no ha podido enviar el trabajo porque la aplicación ha superado los límites de capacidad máxima configurados. Aumente los límites de capacidad máxima de la aplicación.

Error: se ha superado la capacidad máxima configurada. Inténtelo de nuevo más tarde.

Este error indica que EMR Serverless no ha podido iniciar un nuevo trabajo porque la aplicación ha superado los límites de capacidad máxima configurados. Aumente los límites de capacidad máxima de la aplicación.

Error: se ha denegado el acceso a S3. Compruebe los permisos de acceso a S3 del rol de ejecución del trabajo en los recursos de S3 necesarios.

Este error indica que su trabajo no tiene acceso a sus recursos de S3. Compruebe que el rol de ejecución del trabajo tenga permiso para acceder a los recursos de S3 que debe usar el trabajo. Para obtener más información sobre las funciones en tiempo de ejecución, consulteFunciones de tiempo de ejecución de trabajos para Amazon EMR Serverless.

Error ModuleNotFoundError: No hay ningún módulo con nombre<module>. Consulte la guía del usuario sobre cómo utilizar las bibliotecas de Python con EMR Serverless.

Este error indica que no había un módulo de Python disponible para el trabajo de Spark. Compruebe que las bibliotecas de Python dependientes estén disponibles para el trabajo. Para obtener más información sobre cómo empaquetar las bibliotecas de Python, consulteUso de bibliotecas de Python con EMR Serverless.

Error: no se pudo asumir la función de ejecución <role name>porque no existe o no está configurada con la relación de confianza requerida.

Este error indica que el rol de ejecución del trabajo que especificó para el trabajo no existe o que el rol no tiene una relación de confianza para los permisos de EMR Serverless. Para comprobar que el IAM rol existe y validar que has configurado correctamente la política de confianza del rol, consulta las instrucciones que aparecen enFunciones de tiempo de ejecución de trabajos para Amazon EMR Serverless.

Ejecute cargas de trabajo interactivas con EMR Serverless a través de Studio EMR

Información general

Una aplicación interactiva es una aplicación EMR sin servidor que tiene habilitadas las capacidades interactivas. Con las aplicaciones interactivas Amazon EMR Serverless, puede ejecutar cargas de trabajo interactivas con los cuadernos de Jupyter que se administran en Amazon Studio. EMR Esto ayuda a los ingenieros de datos, científicos de datos y analistas de datos a utilizar EMR Studio para ejecutar análisis interactivos con conjuntos de datos en almacenes de datos como Amazon S3 y Amazon DynamoDB.

Los casos de uso de aplicaciones interactivas en EMR Serverless incluyen los siguientes:

- Los ingenieros de datos utilizan la IDE experiencia de EMR Studio para crear un ETL script. El script ingiere datos de las instalaciones, los transforma para su análisis y los almacena en Amazon S3.
- Los científicos de datos utilizan cuadernos para explorar conjuntos de datos y entrenar modelos de aprendizaje automático (ML) para detectar anomalías en los conjuntos de datos.
- Los analistas de datos exploran los conjuntos de datos y crean scripts que generan informes diarios para actualizar aplicaciones, como los cuadros de mando empresariales.

Requisitos previos

Para utilizar cargas de trabajo interactivas con EMR Serverless, debe cumplir los siguientes requisitos:

- EMRLas aplicaciones interactivas sin servidor son compatibles con Amazon EMR 6.14.0 y versiones posteriores.
- Para acceder a su aplicación interactiva, ejecutar las cargas de trabajo que envíe y ejecutar cuadernos interactivos desde EMR Studio, necesita permisos y funciones específicos. Para obtener más información, consulte Permisos necesarios para las cargas de trabajo interactivas.

Permisos necesarios para las cargas de trabajo interactivas

Además de los permisos básicos necesarios para acceder a EMR Serverless, debe configurar permisos adicionales para su IAM identidad o función:

Para acceder a su aplicación interactiva

Configura los permisos de usuario y de Workspace para EMR Studio. Para obtener más información, consulte Configurar los permisos de usuario de EMR Studio en la Amazon EMR Management Guide.

Para ejecutar las cargas de trabajo que envíe con Serverless EMR

Configure un rol de ejecución de tareas. Para obtener más información, consulte Cree un rol de ejecución de tareas.

Para ejecutar los cuadernos interactivos desde Studio EMR

Añada los siguientes permisos adicionales a la IAM política para los usuarios de Studio:

- **`emr-serverless:AccessInteractiveEndpoints`**- Otorga permiso para acceder a la aplicación interactiva que especifique y conectarse a ella`Resource`. Este permiso es necesario para adjuntarlo a una aplicación EMR sin servidor desde un espacio de trabajo de EMR Studio.
- **`iam:PassRole`**- Otorga permiso para acceder a la función de IAM ejecución que va a utilizar al adjuntarla a una aplicación. Se requiere el `PassRole` permiso correspondiente para conectarse a una aplicación EMR sin servidor desde un espacio de trabajo de EMR Studio.

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

Configuración de aplicaciones interactivas

Siga los siguientes pasos de alto nivel para crear una aplicación EMR sin servidor con funciones interactivas de Amazon EMR Studio en el AWS Management Console.

1. Siga los pasos que se indican Introducción a Amazon EMR Serverless a continuación para crear una aplicación.
2. A continuación, inicie un espacio de trabajo desde EMR Studio y conéctelo a una aplicación EMR sin servidor como opción de procesamiento. Para obtener más información, consulte la pestaña Carga de trabajo interactiva en el paso 2 de la documentación de introducción a EMR Serverless.

Al adjuntar una aplicación a un espacio de trabajo de Studio, el inicio de la aplicación se activa automáticamente si aún no se está ejecutando. También puedes preiniciar la aplicación y tenerla lista antes de adjuntarla al espacio de trabajo.

Consideraciones sobre las aplicaciones interactivas

- EMRLas aplicaciones interactivas sin servidor son compatibles con Amazon EMR 6.14.0 y versiones posteriores.
- EMRStudio es el único cliente que está integrado con las aplicaciones interactivas EMR sin servidor. Las siguientes funciones de EMR Studio no son compatibles con las aplicaciones interactivas EMR sin servidor: Workspace Collaboration, SQL Explorer y ejecución programática de cuadernos.
- Las aplicaciones interactivas solo son compatibles con el motor Spark.
- Las aplicaciones interactivas son compatibles con los núcleos de Python 3 PySpark y Spark Scala.
- Puedes ejecutar hasta 25 cuadernos simultáneos en una sola aplicación interactiva.

- No hay un punto final o una API interfaz que admita los cuadernos Jupyter autohospedados con aplicaciones interactivas.
- Para una experiencia de inicio optimizada, le recomendamos que configure la capacidad preinicializada para los controladores y ejecutores y que inicie previamente la aplicación. Al iniciar previamente la aplicación, asegúrate de que esté lista cuando quieras adjuntarla a tu espacio de trabajo.

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

- De forma predeterminada, `autoStopConfig` está habilitada para las aplicaciones. Esto cierra la aplicación después de 30 minutos de inactividad. Puede cambiar esta configuración como parte de su `create-application` `update-application` solicitud.
- Cuando utilice una aplicación interactiva, le recomendamos que configure una capacidad preinicializada de núcleos, controladores y ejecutores para ejecutar sus ordenadores portátiles. Cada sesión interactiva de Spark requiere un núcleo y un controlador, por lo que EMR Serverless mantiene un servidor de núcleo preinicializado para cada controlador preinicializado. De forma predeterminada, EMR Serverless mantiene la capacidad preinicializada de un servidor del núcleo en toda la aplicación, aunque no especifiques ninguna capacidad preinicializada para los controladores. Cada servidor del núcleo utiliza 4 v CPU y 16 GB de memoria. Para obtener información sobre los precios actuales, consulta la página de EMRprecios de Amazon.
- Debe tener una cuota de CPU servicio V suficiente en su Cuenta de AWS para ejecutar cargas de trabajo interactivas. Si no ejecuta cargas de trabajo compatibles con Lake Formation, le recomendamos que utilice al menos 24 v. CPU Si lo hace, le recomendamos que utilice al menos 28 v. CPU
- EMRServerless cierra automáticamente los núcleos de los cuadernos si han estado inactivos durante más de 60 minutos. EMRServerless calcula el tiempo de inactividad del núcleo a partir de la última actividad completada durante la sesión del bloc de notas. Actualmente, no se puede modificar la configuración de tiempo de espera de inactividad del núcleo.
- Para habilitar Lake Formation con cargas de trabajo interactivas, `spark.emr-serverless.lakeformation.enabled` defina `true` la configuración en la `spark-defaults` clasificación del `runtime-configuration` objeto al crear una aplicación EMR sin servidor. Para obtener más información sobre cómo habilitar Lake Formation en EMR Serverless, consulta Cómo habilitar Lake Formation en Amazon EMR.

Ejecute cargas de trabajo interactivas con EMR Serverless a través de un terminal Apache Livy

Con las EMR versiones 6.14.0 y posteriores de Amazon, puede crear y habilitar un punto de conexión Apache Livy al mismo tiempo crear una aplicación EMR sin servidor y ejecutar cargas de trabajo interactivas a través de sus blocs de notas autohospedados o con un cliente personalizado. Un terminal Apache Livy ofrece las siguientes ventajas:

- Puedes conectarte de forma segura a un terminal Apache Livy a través de los cuadernos de Jupyter y gestionar las cargas de trabajo de Apache Spark con la interfaz de Apache Livy. REST
- Utilice las REST API operaciones de Apache Livy para aplicaciones web interactivas que utilizan datos de las cargas de trabajo de Apache Spark.

Requisitos previos

Para utilizar un terminal Apache Livy con EMR Serverless, debe cumplir los siguientes requisitos:

- Complete los pasos que se indican en Introducción a Amazon EMR Serverless.
- Para ejecutar cargas de trabajo interactivas a través de los puntos de conexión de Apache Livy, necesita ciertos permisos y funciones. Para obtener más información, consulte Permisos necesarios para las cargas de trabajo interactivas.

Permisos necesarios

Además de los permisos necesarios para acceder a EMR Serverless, también debe añadir los siguientes permisos a su IAM función para acceder a un punto final de Apache Livy y ejecutar aplicaciones:

- `emr-serverless:AccessLivyEndpoints`— concede permiso para acceder y conectarse a la aplicación habilitada para Livy que usted especifique como. `Resource` Necesita este permiso para ejecutar REST API las operaciones disponibles en el punto final de Apache Livy.
- `iam:PassRole`— concede permiso para acceder al rol de IAM ejecución al crear la sesión de Apache Livy. EMRServerless utilizará esta función para ejecutar sus cargas de trabajo.
- `emr-serverless:GetDashboardForJobRun`— concede permiso para generar los enlaces a la interfaz de usuario y al registro de controladores de Spark Live y proporciona acceso a los registros como parte de los resultados de la sesión de 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/*"
        }
    ]
}
```

Introducción

1. Para crear una aplicación compatible con Apache Livy, ejecuta el siguiente comando.

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

2. Una vez que EMR Serverless haya creado la aplicación, iníciela para que el punto final de Apache Livy esté disponible.

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

```
 --application-id application-id
```

Utilice el siguiente comando para comprobar el estado de su aplicación. Una vez que el estado pase a serSTARTED, podrá acceder al punto final de Apache Livy.

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

3. Utilice lo siguiente URL para acceder al punto final:

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

Una vez que el punto final esté listo, puede enviar las cargas de trabajo en función de su caso de uso. Debe firmar todas las solicitudes que se envíen al punto final con el SIGv4 protocolo y pasar un encabezado de autorización. Puede usar los siguientes métodos para ejecutar cargas de trabajo:

- HTTPcliente: debe enviar sus API operaciones de punto final de Apache Livy con un cliente personalizadoHTTP.
- Núcleo de Sparkmagic: debe ejecutar el núcleo de Sparkmagic de forma local y enviar consultas interactivas con los cuadernos de Jupyter.

HTTPclientes

Para crear una sesión de Apache Livy, debe introducir `emr-serverless.session.executionRoleArn` el `conf` parámetro del cuerpo de la solicitud. El siguiente ejemplo es un ejemplo de `POST /sessions` solicitud.

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

En la siguiente tabla se describen todas las API operaciones de Apache Livy disponibles.

APIoperación	Descripción
GET/sesiones	Devuelve una lista de todas las sesiones interactivas activas.
POST/sesiones	Crea una nueva sesión interactiva mediante spark o pyspark.
GET/sessions/ <*sessionId* >	Devuelve la información de la sesión.
GET/sessions/ <*sessionId* /sesiones/estado	Devuelve el estado de la sesión.
DELETE/sessiones/ <*sessionId* >	Detiene y elimina la sesión.
GET/sessions/ <*sessionId* /sesiones/declaraciones	Devuelve todas las declaraciones de una sesión.
POST/sessions/ <*sessionId* /sesiones/declaraciones	Ejecuta una declaración en una sesión.
GET/sessions/ <*sessionId* /sesiones/declaraciones/<*statementId* >	Devuelve los detalles de la declaración especificada en una sesión.
POST/sessions/ <*sessionId* /sesiones/declaraciones/<*statementId* >/declaraciones/cancelar	Cancela la declaración especificada en esta sesión.

Envío de solicitudes al punto final de Apache Livy

También puede enviar solicitudes directamente al punto final de Apache Livy desde un HTTP cliente. De este modo, podrá ejecutar código de forma remota para sus casos de uso fuera de un bloc de notas.

Antes de empezar a enviar solicitudes al punto final, asegúrate de haber instalado las siguientes bibliotecas:

```
pip3 install botocore awscrt requests
```

El siguiente es un ejemplo de script de Python para enviar HTTP solicitudes directamente a un punto final:

```
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())
```

Núcleo de Sparkmagic

Antes de instalar Sparkmagic, asegúrese de haber configurado AWS credenciales en la instancia en la que desea instalar sparkmagic

1. Instale sparkmagic siguiendo los pasos de instalación. Tenga en cuenta que solo necesita realizar los cuatro primeros pasos.
2. El núcleo de Sparkmagic admite autenticadores personalizados, por lo que puede integrar un autenticador con el núcleo de Sparkmagic para que todas las solicitudes estén firmadas. SIGv4
3. Instale el autenticador personalizado ServerlessEMR.

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

4. Ahora proporciona la ruta al autenticador personalizado y al punto final de Apache Livy URL en el archivo json de configuración de sparkmagic. Use el siguiente comando para abrir el archivo de configuración.

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

A continuación se muestra un archivo de muestra `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 Jupyter lab. Debe usar la autenticación personalizada que configuraste en el último paso.
6. A continuación, puede ejecutar los siguientes comandos del bloc de notas y su código para empezar.

```
%%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 instrucción llama a cada una de las API operaciones de Apache Livy a través del punto final de Apache Livy configurado. URL A continuación, puede escribir las instrucciones de acuerdo con su caso de uso.

Consideraciones

Tenga en cuenta las siguientes consideraciones al ejecutar cargas de trabajo interactivas a través de los puntos finales de Apache Livy.

- EMRServerless mantiene el aislamiento a nivel de sesión mediante el principal que realiza la llamada. El principal que llama y crea la sesión es el único que puede acceder a esa sesión. Para un aislamiento más detallado, puede configurar una identidad de origen al asumir las credenciales. En este caso, EMR Serverless aplica el aislamiento a nivel de sesión en función de la identidad principal de la persona que llama y de la fuente. Para obtener más información sobre la identidad de la fuente, consulte Supervisar y controlar las acciones realizadas con los roles asumidos.
- Los puntos finales de Apache Livy son compatibles con las versiones 6.14.0 y superiores de EMR Serverless.

- Los puntos finales de Apache Livy solo son compatibles con el motor Apache Spark.
- Los puntos finales de Apache Livy son compatibles con Scala Spark y. PySpark
- De forma predeterminada, `autoStopConfig` está habilitada en sus aplicaciones. Esto significa que las aplicaciones se cierran después de 15 minutos de estar inactivas. Puede cambiar esta configuración como parte de su `create-application` `update-application` solicitud.
- Puede ejecutar hasta 25 sesiones simultáneas en una sola aplicación habilitada para terminales Apache Livy.
- Para obtener la mejor experiencia de inicio, le recomendamos que configure la capacidad preinicializada para los controladores y los ejecutores.
- Debe iniciar la aplicación manualmente antes de conectarse al punto final de Apache Livy.
- Debe tener una cuota de CPU servicio v suficiente en su Cuenta de AWS para ejecutar cargas de trabajo interactivas con el punto final Apache Livy. Se recomienda utilizar al menos 24 V. CPU
- El tiempo de espera predeterminado de la sesión de Apache Livy es de 1 hora. Si no ejecuta las sentencias durante una hora, Apache Livy borra la sesión y libera el controlador y los ejecutores. No puede cambiar esta configuración.
- Solo las sesiones activas pueden interactuar con un punto final de Apache Livy. Una vez que la sesión finalice, se cancele o termine, no podrá acceder a ella a través del punto final de Apache Livy.

Registro y monitorización

La supervisión es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de las aplicaciones y los trabajos EMR sin servidor. Debe recopilar datos de supervisión de todas las partes de sus soluciones EMR sin servidor para poder depurar más fácilmente un error multipunto en caso de que se produzca.

Temas

Almacenamiento de registros

Para supervisar el progreso de su trabajo en EMR Serverless y solucionar los errores en los trabajos, puede elegir la forma en que EMR Serverless almacena y publica los registros de las aplicaciones. Al enviar una ejecución de trabajo, puede especificar el almacenamiento gestionado, Amazon S3 y Amazon CloudWatch como opciones de registro.

Con CloudWatch, puede especificar los tipos de registro y las ubicaciones de registro que quiere usar, o bien aceptar los tipos y ubicaciones predeterminados. Para obtener más información sobre CloudWatch los registros, consultethe section called “Amazon CloudWatch”. Con el almacenamiento gestionado y el registro de S3, la siguiente tabla muestra las ubicaciones de registro y la disponibilidad de la interfaz de usuario que puede esperar si elige almacenamiento gestionado, depósitos de Amazon S3 o ambos.

Opción	Registros de eventos	Registros de contenedor	Interfaz de usuario de aplicación
Almacenamiento gestionado	Almacenado en almacenamiento gestionado	Almacenado en un almacenamiento gestionado	Compatible

Opción	Registros de eventos	Registros de contenedor	Interfaz de usuario de aplicación
Tanto almacenamiento gestionado como depósito S3	Almacenado en ambos lugares	Almacenado en un depósito S3	Compatible
Bucket de Amazon S3	Almacenado en un depósito S3	Almacenado en un depósito S3	No se admite [1]

[1] Le recomendamos que mantenga seleccionada la opción Almacenamiento gestionado. De lo contrario, no podrá utilizar la aplicación integradaUIs.

Registro para sistemas EMR sin servidor con almacenamiento gestionado

De forma predeterminada, EMR Serverless almacena los registros de las aplicaciones de forma segura en el almacenamiento EMR gestionado por Amazon durante un máximo de 30 días.

> **Note**
>
> Si desactivas la opción predeterminada, Amazon no EMR podrá solucionar los problemas de tus trabajos en tu nombre.

Para desactivar esta opción desde EMR Studio, deselecciona la opción Permitir AWS para conservar los registros durante 30 días, seleccione la casilla de verificación en la sección Configuración adicional de la página Enviar trabajo.

Para desactivar esta opción desde el AWS CLI, utilice la `managedPersistenceMonitoringConfiguration` configuración cuando envíe una ejecución de trabajo.

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

Registro para aplicaciones EMR sin servidor con buckets de Amazon S3

Antes de que sus trabajos puedan enviar datos de registro a Amazon S3, debe incluir los siguientes permisos en la política de permisos del rol de ejecución del trabajo. *DOC-EXAMPLE-BUCKET-LOGGING*Sustitúyalos por el nombre de su depósito de registro.

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

Para configurar un bucket de Amazon S3 para almacenar los registros del AWS CLI, utilice la `s3MonitoringConfiguration` configuración al iniciar la ejecución de un trabajo. Para ello, incluya lo siguiente `--configuration-overrides` en la configuración.

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

Para los trabajos por lotes que no tienen habilitados los reintentos, EMR Serverless envía los registros a la siguiente ruta:

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

EMRLas versiones 7.1.0 y posteriores sin servidor admiten reintentos para trabajos de streaming y trabajos por lotes. Si ejecuta un trabajo con los reintentos habilitados, EMR Serverless agrega

automáticamente un número de intento al prefijo de la ruta del registro para que pueda distinguir y rastrear mejor los registros.

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

Cómo iniciar sesión EMR sin servidor con Amazon CloudWatch

Cuando envías un trabajo a una aplicación EMR sin servidor, puedes elegir Amazon CloudWatch como opción para almacenar los registros de tu aplicación. Esto le permite utilizar CloudWatch funciones de análisis de CloudWatch registros, como Logs Insights y Live Tail. También puede transmitir registros desde CloudWatch otros sistemas, por ejemplo, OpenSearch para su posterior análisis.

EMRServerless proporciona un registro en tiempo real de los registros de los controladores. Puede ver los registros en tiempo real con la función de seguimiento CloudWatch en tiempo real o mediante comandos de CloudWatch CLI seguimiento.

De forma predeterminada, el CloudWatch registro está deshabilitado en EMR Serverless. Para habilitarlo, consulte la configuración enAWS CLI.

> **Note**
>
> Amazon CloudWatch publica los registros en tiempo real, por lo que obtiene más recursos de los trabajadores. Si eliges una capacidad laboral baja, el impacto en el tiempo de ejecución de tu trabajo podría aumentar. Si habilita el CloudWatch registro, le recomendamos que elija una mayor capacidad de trabajo. También es posible que la publicación del registro se reduzca si la tasa de transacciones por segundo (TPS) es demasiado baja. `PutLogEvents` La configuración de CloudWatch limitación es global para todos los servicios, incluido Serverless. EMR Para obtener más información, consulte ¿Cómo puedo determinar la limitación en mis registros? CloudWatch en AWS re:publicar.

Permisos necesarios para iniciar sesión con CloudWatch

Para que tus trabajos puedan enviar datos de registro a Amazon CloudWatch, debes incluir los siguientes permisos en la política de permisos del rol de ejecución del trabajo.

```
{
```

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

AWS CLI

Para configurar Amazon CloudWatch para que almacene los registros de EMR Serverless desde el AWS CLI, utilice la `cloudWatchLoggingConfiguration` configuración al iniciar la ejecución de una tarea. Para ello, proporcione las siguientes anulaciones de configuración. Si lo desea, también puede proporcionar un nombre de grupo de registros, un nombre de prefijo de flujo de registro, tipos de registro y una clave de cifrado. ARN

Si no especifica los valores opcionales, CloudWatch publica los registros en un grupo de registros predeterminado`/aws/emr-serverless`, con el flujo `/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`* de registros predeterminado.

EMRLas versiones 7.1.0 y posteriores sin servidor admiten reintentos para trabajos de streaming y trabajos por lotes. Si habilitó los reintentos para un trabajo, EMR Serverless agrega automáticamente un número de intento al prefijo de la ruta del registro, para que pueda distinguir y rastrear mejor los registros.

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

A continuación se muestra la configuración mínima necesaria para activar el CloudWatch registro de Amazon con la configuración predeterminada de EMR Serverless:

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

El siguiente ejemplo muestra todas las configuraciones obligatorias y opcionales que puede especificar al activar Amazon CloudWatch Logging para EMR Serverless. Los `logTypes` valores admitidos también se muestran debajo de este ejemplo.

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

De forma predeterminada, EMR Serverless publica solo los registros stdout y stderr del controlador. CloudWatch Si desea otros registros, puede especificar un rol de contenedor y los tipos de registro correspondientes con el campo. `logTypes`

La siguiente lista muestra los tipos de trabajadores compatibles que puede especificar para la `logTypes` configuración:

Spark

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

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

Hive

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

Registros giratorios

Amazon EMR Serverless puede rotar los registros de aplicaciones y eventos de Spark. La rotación de registros ayuda a solucionar el problema de que los trabajos de larga ejecución generen archivos de registro de gran tamaño que pueden ocupar todo el espacio en el disco. La rotación de los registros le ayuda a ahorrar espacio de almacenamiento en disco y reduce la cantidad de errores en los trabajos, ya que no queda más espacio en el disco.

La rotación de registros está habilitada de forma predeterminada y solo está disponible para los trabajos de Spark.

Registros de eventos de Spark

> **Note**
>
> La rotación del registro de eventos de Spark está disponible en todas las etiquetas EMR de lanzamiento de Amazon.

En lugar de generar un único archivo de registro de eventos, EMR Serverless rota el registro de eventos a intervalos de tiempo regulares y elimina los archivos de registro de eventos más antiguos. La rotación de los registros no afecta a los registros cargados en el bucket de S3.

Registros de aplicaciones de Spark

> **Note**
>
> La rotación del registro de aplicaciones de Spark está disponible en todas las etiquetas EMR de lanzamiento de Amazon.

EMRServerless también rota los registros de las aplicaciones de Spark para guardarlos en los controladores y ejecutores, como `stdout` los archivos. `stderr` Para acceder a los archivos de

registro más recientes, selecciona los enlaces de registro de Studio mediante los enlaces del servidor de historial de Spark y de la interfaz de usuario en vivo. Los archivos de registro son las versiones truncadas de los registros más recientes. Para ver los registros rotados más antiguos, debe especificar una ubicación de Amazon S3 al almacenar los registros. Consulte Logging for EMR Serverless with Amazon S3 buckets para obtener más información.

Puede encontrar los archivos de registro más recientes en la siguiente ubicación. EMRServerless actualiza los archivos cada 15 segundos. Estos archivos pueden oscilar entre 0 MB y 128 MB.

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

La siguiente ubicación contiene los archivos rotados más antiguos. Cada archivo tiene un tamaño de 128 MB.

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

El mismo comportamiento se aplica también a los ejecutores de Spark. Este cambio solo se aplica al registro de S3. La rotación de registros no introduce ningún cambio en las transmisiones de registros subidas a Amazon CloudWatch.

EMRLas versiones 7.1.0 y posteriores sin servidor admiten reintentos para trabajos de streaming y por lotes. Si ha habilitado los reintentos con su trabajo, EMR Serverless añade un prefijo a la ruta de registro de dichos trabajos para que pueda rastrear y distinguir mejor los registros entre sí. Esta ruta contiene todos los registros rotados.

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

Cifrar registros

Cifrar registros EMR sin servidor con almacenamiento gestionado

Para cifrar los registros del almacenamiento gestionado con su propia KMS clave, utilice la `managedPersistenceMonitoringConfiguration` configuración al enviar una ejecución de tareas.

```
{
```

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

Cifrado de registros EMR sin servidor con buckets de Amazon S3

Para cifrar los registros de su bucket de Amazon S3 con su propia KMS clave, utilice la `s3MonitoringConfiguration` configuración cuando envíe una ejecución de trabajo.

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

Cifrado de registros EMR sin servidor con Amazon CloudWatch

Para cifrar los registros en Amazon CloudWatch con tu propia KMS clave, usa la `cloudWatchLoggingConfiguration` configuración cuando envíes una ejecución de trabajo.

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

Permisos necesarios para el cifrado de registros

En esta sección

- Permisos de clave de cifrado para Amazon CloudWatch

Permisos de usuario necesarios

El usuario que envía el trabajo o ve los registros o la aplicación UIs debe tener permisos para usar la clave. Puede especificar los permisos en la política KMS clave o en la IAM política del usuario, grupo o rol. Si el usuario que envía el trabajo carece de los permisos KMS clave, EMR Serverless rechaza el envío de la ejecución del trabajo.

Ejemplo de política clave

La siguiente política clave proporciona los permisos `kms:GenerateDataKey` y`kms:Decrypt`:

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

Ejemplo IAM de política

La siguiente IAM política proporciona los permisos para `kms:GenerateDataKey` y`kms:Decrypt`:

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

Para iniciar la interfaz de usuario de Spark o Tez, debes conceder a tus usuarios, grupos o roles los permisos de acceso `emr-serverless:GetDashboardForJobRun` API siguientes:

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

Permisos de clave de cifrado para Amazon S3 y almacenamiento gestionado

Al cifrar los registros con su propia clave de cifrado, ya sea en el almacenamiento gestionado o en los depósitos de S3, debe configurar los permisos de KMS clave de la siguiente manera.

El `emr-serverless.amazonaws.com` principal debe tener los siguientes permisos en la política de la KMS clave:

```
{
    "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 práctica recomendada de seguridad, le recomendamos que añada una clave de `aws:SourceArn` condición a la política de KMS claves. La clave de condición IAM global

`aws:SourceArn` ayuda a garantizar que EMR Serverless utilice la KMS clave solo para una aplicaciónARN.

El rol de ejecución del trabajo debe tener los siguientes permisos en su IAM política:

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

Permisos de clave de cifrado para Amazon CloudWatch

Para asociar la KMS clave ARN a su grupo de registros, utilice la siguiente IAM política para el rol de ejecución del trabajo.

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

Configura la política KMS clave para conceder KMS permisos a Amazon CloudWatch:

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

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

Configuración de las propiedades de Apache Log4j2 para Amazon Serverless EMR

Esta página describe cómo configurar las propiedades personalizadas de Apache Log4j 2.x para trabajos sin servidor en. EMR `StartJobRun` Si desea configurar las clasificaciones de Log4j a nivel de aplicación, consulte. Configuración de aplicaciones predeterminada para Serverless EMR

Configurar las propiedades de Spark Log4j2 para Amazon Serverless EMR

Con las EMR versiones 6.8.0 y posteriores de Amazon, puede personalizar las propiedades de Apache Log4j 2.x para especificar configuraciones de registro detalladas. Esto simplifica la solución de problemas de tus trabajos de Spark en Serverless. EMR Para configurar estas propiedades, usa las clasificaciones `spark-driver-log4j2` y`spark-executor-log4j2`.

Temas

- Clasificaciones Log4j2 para Spark
- Ejemplo de configuración de Log4j2 para Spark
- Log4j2 en ejemplos de trabajos de Spark
- Consideraciones sobre Log4j2 para Spark

Clasificaciones Log4j2 para Spark

Para personalizar las configuraciones de registro de Spark, usa las siguientes clasificaciones con. `applicationConfiguration` Para configurar las propiedades de Log4j 2.x, usa lo siguiente. `properties`

`spark-driver-log4j2`

Esta clasificación establece los valores del controlador en el `log4j2.properties` archivo.

`spark-executor-log4j2`

Esta clasificación establece los valores del `log4j2.properties` archivo para el ejecutor.

Ejemplo de configuración de Log4j2 para Spark

El siguiente ejemplo muestra cómo enviar un trabajo de Spark para personalizar las configuraciones de Log4j2 para el controlador y el ejecutor de Spark. `applicationConfiguration`

Para configurar las clasificaciones de Log4j a nivel de aplicación y no al enviar el trabajo, consulte. Configuración de aplicaciones predeterminada para Serverless EMR

```
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 en ejemplos de trabajos de Spark

Los siguientes ejemplos de código muestran cómo crear una aplicación Spark mientras se inicializa una configuración Log4j2 personalizada para la aplicación.

Python

Example - Uso de Log4j2 para un trabajo de Spark con 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 Log4j2 para el controlador cuando ejecutas un trabajo de Spark, puedes usar la siguiente configuración:

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

Scala

Example - Uso de Log4j2 para un trabajo de Spark con 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 Log4j2 para el controlador cuando ejecutas un trabajo de Spark, puedes usar la siguiente configuración:

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

Consideraciones sobre Log4j2 para Spark

Las siguientes propiedades de Log4j2.x no se pueden configurar para los procesos de Spark:

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

Para obtener información detallada sobre las propiedades de Log4j2.x que puede configurar, consulte el archivo en. `log4j2.properties.template` GitHub

Supervisión sin servidor EMR

En esta sección se describen las formas en que puede supervisar sus aplicaciones y trabajos de Amazon EMR Serverless.

Temas

- Supervisión de aplicaciones y EMR trabajos sin servidor
- Supervisa las métricas de Spark con Amazon Managed Service para Prometheus
- EMRMétricas de uso sin servidor

Supervisión de aplicaciones y EMR trabajos sin servidor

Con CloudWatch las métricas de Amazon para EMR Serverless, puede recibir CloudWatch métricas de 1 minuto y acceder a los CloudWatch paneles para ver las near-real-time operaciones y el rendimiento de sus EMR aplicaciones sin servidor.

EMRServerless envía métricas cada minuto. CloudWatch EMRServerless emite estas métricas a nivel de aplicación, así como a nivel de puesto, tipo de trabajador y niveles. capacity-allocation-type

Para empezar, usa la plantilla de CloudWatch panel de control de EMR Serverless que se proporciona en el repositorio de EMR GitHub Serverless e impleméntala.

> **Note**
>
> EMRLas cargas de trabajo interactivas sin servidor solo tienen habilitada la supervisión a nivel de aplicación y tienen una nueva dimensión de tipo trabajador. `Spark_Kernel` Para supervisar y depurar tus cargas de trabajo interactivas, puedes ver los registros y la interfaz de usuario de Apache Spark desde tu espacio de trabajo de Studio. EMR

La siguiente tabla describe las dimensiones EMR sin servidor disponibles en el espacio de nombres. `AWS/EMRServerless`

Dimensiones de las métricas de Serverless EMR

Dimensión	Descripción	
`ApplicationId`	Filtros para todas las métricas de una aplicación EMR sin servidor.	
`JobId`	Filtra todas las métricas de la ejecución de un trabajo EMR sin servidor.	
`WorkerType`	Filtra todas las métricas de un tipo de trabajador determina do. Por ejemplo, puedes filtrar por `SPARK_DRIVER` y	

Dimensión	Descripción	
	`SPARK_EXECUTORS` para los trabajos de Spark.	
`CapacityAllocationType`	Filtra todas las métricas de un tipo de asignación de capacidad determinado. Por ejemplo, puede filtrar `PreInitCapacity` por la capacidad preinicializada y `OnDemandCapacity` por todo lo demás.	

Supervisión a nivel de aplicación

Puedes monitorizar el uso de la capacidad a nivel de aplicación EMR sin servidor con CloudWatch las métricas de Amazon. También puede configurar una vista única para monitorear el uso de la capacidad de las aplicaciones en un CloudWatch panel de control.

EMRMétricas de aplicaciones sin servidor

Métrica	Descripción	Dimensión principal	Dimensión secundaria
`CPUAllocated`	El número total de vCPUs asignados.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`IdleWorkerCount`	El número total de trabajadores inactivos .	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`MaxCPUAllowed`	El máximo CPU permitido para la aplicación.	`ApplicationId`	N/A

Métrica	Descripción	Dimensión principal	Dimensión secundaria
MaxMemoryAllowed	Memoria máxima en GB permitida para la aplicación.	ApplicationId	N/A
MaxStorageAllowed	El almacenamiento máximo en GB permitido para la aplicación.	ApplicationId	N/A
MemoryAllocated	La memoria total en GB asignada.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
PendingCreationWorkerCount	El número total de trabajadores pendientes de creación.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
RunningWorkerCount	El número total de trabajadores que utiliza la aplicación.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
StorageAllocated	El almacenamiento total en disco asignado en GB.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
TotalWorkerCount	El número total de trabajadores disponibles.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType

Supervisión a nivel de puesto

Amazon EMR Serverless envía las siguientes métricas de nivel de trabajo a Amazon CloudWatch cada minuto. Puede ver los valores métricos de las ejecuciones de tareas agregadas por estado de ejecución. La unidad de cada una de las métricas es el recuento.

EMRMétricas a nivel de trabajo sin servidor

Métrica	Descripción	Dimensión principal
`SubmittedJobs`	El número de trabajos en estado Enviado.	`ApplicationId`
`PendingJobs`	El número de trabajos en un estado pendiente.	`ApplicationId`
`ScheduledJobs`	El número de trabajos en un estado programado.	`ApplicationId`
`RunningJobs`	El número de trabajos en estado En ejecución.	`ApplicationId`
`SuccessJobs`	El número de trabajos en un estado de éxito.	`ApplicationId`
`FailedJobs`	El número de trabajos en un estado fallido.	`ApplicationId`
`CancellingJobs`	El número de trabajos en estado de cancelación.	`ApplicationId`
`CancelledJobs`	El número de trabajos en un estado cancelado.	`ApplicationId`

Puede monitorear las métricas específicas del motor para los trabajos EMR sin servidor en ejecución y completados con una aplicación específica del motor. UIs Al ver la interfaz de usuario de un trabajo en ejecución, verá la interfaz de usuario de la aplicación en tiempo real con actualizaciones en tiempo real. Cuando ves la interfaz de usuario de un trabajo completado, ves la interfaz de usuario persistente de la aplicación.

Trabajos en ejecución

Para tus trabajos EMR sin servidor en ejecución, puedes ver una interfaz en tiempo real que proporciona métricas específicas del motor. Puede utilizar la interfaz de usuario de Apache Spark o la interfaz de usuario de Hive Tez para supervisar y depurar sus trabajos. Para acceder a ellasUIs, usa la consola de EMR Studio o solicita un URL terminal seguro con el AWS Command Line Interface.

Trabajos completados

Para los trabajos EMR sin servidor que hayas completado, puedes usar el servidor de historial de Spark o la interfaz de usuario persistente de Hive Tez para ver los detalles de los trabajos, las etapas, las tareas y las métricas de los trabajos ejecutados en Spark o Hive. Para acceder a ellosUIs, usa la consola de EMR Studio o solicita un terminal seguro URL con el AWS Command Line Interface.

Supervisión a nivel de trabajador de Job

Amazon EMR Serverless envía a Amazon las siguientes métricas a nivel de trabajador laboral que están disponibles en el espacio de `AWS/EMRServerless` nombres y el grupo de `Job Worker Metrics` métricas. CloudWatch EMRServerless recopila puntos de datos de los trabajadores individuales durante la ejecución de los trabajos a nivel de puesto, tipo de trabajador y nivel. capacity-allocation-type Puede utilizarlos `ApplicationId` como una dimensión para supervisar varios trabajos que pertenezcan a la misma aplicación.

EMRMétricas a nivel de trabajador sin servidor

Métrica	Descripción	Unidad	Dimensión principal	Dimensión secundaria
`WorkerCpuAllocated`	El número total de CPU núcleos v asignados a los trabajadores en una ejecución de tareas.	Ninguna	`JobId`	`ApplicationId` , `WorkerType` , y `CapacityAllocationType`
`WorkerCpuUsed`	El número total de CPU núcleos v utilizados por	Ninguna	`JobId`	`ApplicationId` , `WorkerType` ,

Métrica	Descripción	Unidad	Dimensión principal	Dimensión secundaria
	los trabajadores en una ejecución de trabajo.			y `CapacityAllocationType`
`WorkerMemoryAllocated`	La memoria total en GB asignada a los trabajadores en una ejecución de tareas.	Gigabytes (GB)	`JobId`	`ApplicationId`, `WorkerType`, y `CapacityAllocationType`
`WorkerMemoryUsed`	Memoria total en GB utilizada por los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`	`ApplicationId`, `WorkerType`, y `CapacityAllocationType`
`WorkerEphemeralStorageAllocated`	El número de bytes de almacenamiento efímero asignados a los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`	`ApplicationId`, `WorkerType`, y `CapacityAllocationType`
`WorkerEphemeralStorageUsed`	La cantidad de bytes de almacenamiento efímero que utilizan los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`	`ApplicationId`, `WorkerType`, y `CapacityAllocationType`

Métrica	Descripción	Unidad	Dimensión principal	Dimensión secundaria
`WorkerStorageReadBytes`	La cantidad de bytes leídos del almacenamiento por los trabajadores en una ejecución de trabajo.	Bytes	`JobId`	`ApplicationId`, `WorkerType`, y `CapacityAllocationType`
`WorkerStorageWriteBytes`	La cantidad de bytes que los trabajadores escriben en el almacenamiento durante una ejecución de trabajo.	Bytes	`JobId`	`ApplicationId`, `WorkerType`, y `CapacityAllocationType`

Los pasos siguientes describen cómo ver los distintos tipos de métricas.

Console

Para acceder a la interfaz de usuario de la aplicación con la consola

1. Dirígete a tu aplicación EMR sin servidor en el EMR estudio siguiendo las instrucciones de Cómo empezar desde la consola.
2. Para ver la aplicación específica del motor UIs y los registros de un trabajo en ejecución:
   a. Elija un trabajo con un `RUNNING` estado.
   b. Seleccione el trabajo en la página de detalles de la solicitud o vaya a la página de detalles del trabajo correspondiente a su trabajo.
   c. En el menú desplegable Mostrar interfaz de usuario, selecciona la interfaz de usuario de Spark o la interfaz de usuario de Hive Tez para navegar hasta la interfaz de usuario de la aplicación correspondiente a tu tipo de trabajo.

d. Para ver los registros del motor de Spark, ve a la pestaña Ejecutores de la interfaz de usuario de Spark y selecciona el enlace Registros correspondiente al conductor. Para ver los registros de Hive Engine, selecciona el enlace Logs correspondiente DAG en la interfaz de usuario de Hive Tez.

3. Para ver la aplicación específica del motor UIs y los registros de un trabajo completado:

a. Elija un trabajo con un `SUCCESS` estado.

b. Seleccione el trabajo en la página de detalles de la solicitud o vaya a la página de detalles del trabajo.

c. En el menú desplegable Mostrar interfaz de usuario, selecciona Spark History Server o Persistent Hive Tez UI para acceder a la interfaz de usuario de la aplicación correspondiente a tu tipo de trabajo.

d. Para ver los registros del motor de Spark, ve a la pestaña Ejecutores de la interfaz de usuario de Spark y selecciona el enlace Registros correspondiente al conductor. Para ver los registros de Hive Engine, selecciona el enlace Logs correspondiente DAG en la interfaz de usuario de Hive Tez.

AWS CLI

Para acceder a la interfaz de usuario de la aplicación con el AWS CLI

- Para generar una URL que pueda utilizar para acceder a la interfaz de usuario de la aplicación tanto para los trabajos en ejecución como para los finalizados, llame al `GetDashboardForJobRun`API.

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

El URL que genere es válido durante una hora.

Supervisa las métricas de Spark con Amazon Managed Service para Prometheus

Con las EMR versiones 7.1.0 y posteriores de Amazon, puede integrar EMR Serverless con Amazon Managed Service for Prometheus para recopilar métricas de Apache Spark para trabajos

y aplicaciones sin servidor. EMR Esta integración está disponible al enviar un trabajo o crear una solicitud mediante el AWS consola, EMR Serverless API o AWS CLI.

Requisitos previos

Antes de poder enviar tus métricas de Spark a Amazon Managed Service for Prometheus, debes cumplir los siguientes requisitos previos.

- Cree un espacio de trabajo de Amazon Managed Service para Prometheus. Este espacio de trabajo sirve como punto de conexión de ingestión. Anote lo que URL se muestra para Endpoint: escritura remota. URL Deberá especificarlo URL cuando cree su aplicación EMR sin servidor.
- Para conceder acceso a sus trabajos a Amazon Managed Service for Prometheus con fines de supervisión, añada la siguiente política a su función de ejecución de trabajos.

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

Configuración

Para utilizar el AWS consola para crear una aplicación que esté integrada con Amazon Managed Service for Prometheus

1. Consulte Introducción a Amazon EMR Serverless para crear una aplicación.
2. Mientras crea una aplicación, elija Usar ajustes personalizados y, a continuación, configure la aplicación especificando la información en los campos que desee configurar.
3. En Registros y métricas de aplicaciones, selecciona Entregar las métricas del motor a Amazon Managed Service for Prometheus y, a continuación, especifica tu escritura remota. URL
4. Especifique cualquier otro ajuste de configuración que desee y, a continuación, seleccione Crear e iniciar la aplicación.

Utilice la AWS CLI o EMR sin servidor API

También puede utilizar el AWS CLI o EMR Serverless API para integrar tu aplicación EMR sin servidor con Amazon Managed Service for Prometheus cuando ejecutes los 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"
        }
    }
}'
```

Incluirlo `prometheusMonitoringConfiguration` en tu comando indica que EMR Serverless debe ejecutar el trabajo de Spark con un agente que recopile las métricas de Spark y las escriba en tu `remoteWriteUrl` punto de conexión para Amazon Managed Service for Prometheus. A

continuación, puede utilizar las métricas de Spark en Amazon Managed Service for Prometheus para la visualización, las alertas y el análisis.

Propiedades de configuración avanzadas

EMRServerless utiliza un componente de Spark denominado `PrometheusServlet` para recopilar las métricas de Spark y traduce los datos de rendimiento en datos compatibles con Amazon Managed Service for Prometheus. De forma predeterminada, EMR Serverless establece los valores predeterminados en Spark y analiza las métricas de los controladores y ejecutores cuando envías un trabajo mediante. `PrometheusMonitoringConfiguration`

En la siguiente tabla se describen todas las propiedades que puedes configurar al enviar un trabajo de Spark que envíe métricas a Amazon Managed Service for Prometheus.

Propiedad de Spark	Valor predeterminado	Descripción
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	La clase que usa Spark para enviar métricas a Amazon Managed Service for Prometheus. Para anular el comportamiento predeterminado, especifica tu propia clase personalizada.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source. JvmSource	La clase que usa Spark para recopilar y enviar métricas cruciales desde la máquina virtual Java subyacente. Para dejar de recopilar JVM métricas, deshabilita esta propiedad configurándola en una cadena vacía, como"". Para anular el comportamiento predeterminado, especifique su propia clase personalizada.

Propiedad de Spark	Valor predeterminado	Descripción
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/metrics/prometheus	El distintivo URL que Amazon Managed Service for Prometheus utiliza para recopilar las métricas del conductor. Para anular el comportamiento predeterminado, especifique su propia ruta. Para dejar de recopilar las métricas de los conductores, deshabilite esta propiedad configurándola en una cadena vacía, como"".
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/executor/prometheus	El distintivo URL que Amazon Managed Service for Prometheus utiliza para recopilar las métricas del ejecutor. Para anular el comportamiento predeterminado, especifique su propia ruta. Para dejar de recopilar las métricas del ejecutor, deshabilite esta propiedad configurándola en una cadena vacía, como. ""

Para obtener más información sobre las métricas de Spark, consulta las métricas de Apache Spark.

Consideraciones y limitaciones

Cuando utilices Amazon Managed Service for Prometheus para recopilar métricas EMR de Serverless, ten en cuenta las siguientes consideraciones y limitaciones.

- El soporte para usar Amazon Managed Service for Prometheus EMR con Serverless solo está disponible en el Regiones de AWS donde Amazon Managed Service for Prometheus está disponible de forma general.
- Ejecutar el agente para recopilar las métricas de Spark en Amazon Managed Service for Prometheus requiere más recursos por parte de los trabajadores. Si eliges un tamaño de trabajador más pequeño, como uno contra un CPU trabajador, el tiempo de ejecución de tu trabajo podría aumentar.
- El soporte para usar Amazon Managed Service for Prometheus EMR con Serverless solo está disponible para las versiones 7.1.0 y superiores de EMR Amazon.

EMRMétricas de uso sin servidor

Puedes usar las métricas CloudWatch de uso de Amazon para proporcionar visibilidad de los recursos que usa tu cuenta. Usa estas métricas para visualizar el uso de tus servicios en CloudWatch gráficos y paneles.

EMRLas métricas de uso sin servidor corresponden a Service Quotas. Puede configurar alarmas que le avisen cuando su uso se acerque a una Service Quota. Para obtener más información, consulte Service Quotas y CloudWatch alarmas de Amazon en la Guía del usuario de Service Quotas.

Para obtener más información sobre las cuotas de servicios EMR sin servidor, consultePuntos finales y cuotas para EMR Serverless.

Métricas de uso de la cuota de servicio para Serverless EMR

EMRServerless publica las siguientes métricas de uso de la cuota de servicio en el `AWS/Usage` espacio de nombres.

Métrica	Descripción
`ResourceCount`	El número total del recurso especificado que se ejecuta en tu cuenta. El recurso se define mediante las dimensiones asociadas a la métrica.

Dimensiones de las métricas de uso de la cuota de servicio EMR sin servidor

Puede usar las siguientes dimensiones para refinar las métricas de uso que publica EMR Serverless.

Dimensión	Valor	Descripción
`Service`	EMRSin servidor	El nombre del Servicio de AWS que contiene el recurso.
`Type`	Recurso	El tipo de entidad sobre la que informa EMR Serverless.
`Resource`	v CPU	El tipo de recurso que EMR Serverless está rastreando.
`Class`	Ninguna	La clase de recurso que EMR Serverless está rastreando.

Automatizar Serverless con EMR Amazon EventBridge

Puede usar… Amazon EventBridge para automatizar su Servicios de AWS y responda automáticamente a los eventos del sistema, como los problemas de disponibilidad de las aplicaciones o los cambios en los recursos. EventBridge ofrece un flujo casi en tiempo real de los eventos del sistema que describen los cambios en su AWS recursos. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas. Con EventBridge, puedes hacer lo siguiente automáticamente:

- Invocar un AWS Lambda función
- Retransmitir un evento a Amazon Kinesis Data Streams
- Active un AWS Step Functions máquina de estado
- Notificar un SNS tema de Amazon o una SQS cola de Amazon

Por ejemplo, cuando lo usas EventBridge con EMR Serverless, puedes activar un AWS Lambda funcionan cuando un ETL trabajo se realiza correctamente o notifica a un SNS tema de Amazon cuando un ETL trabajo fracasa.

EMRServerless emite tres tipos de eventos:

- Eventos de cambio de estado de una aplicación: eventos que emiten cada cambio de estado de una aplicación. Para obtener más información sobre los estados de las aplicaciones, consulteEstados de la aplicación.
- Eventos de cambio de estado de ejecución de una tarea: eventos que emiten cada cambio de estado de una ejecución de tarea. Para obtener más información acerca de ello, consulte Estados de ejecuciones de trabajos.
- Eventos de reintento de ejecución de tareas: eventos que emiten cada reintento de ejecución de una tarea desde las versiones 7.1.0 y posteriores de Amazon EMR Serverless.

Ejemplos de eventos sin servidor EMR EventBridge

Los eventos notificados por EMR Serverless tienen un valor `aws.emr-serverless` asignado `asource`, como en los ejemplos siguientes.

Evento de cambio de estado de la aplicación

El siguiente evento de ejemplo muestra una aplicación en el `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 cambio de estado realizado por Job

El siguiente evento de ejemplo muestra una ejecución de tareas que se mueve de un `SCHEDULED` `RUNNING` estado a otro.

```
{
    "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"
    }
}
```

Job ejecuta un evento de reintento

A continuación se muestra un ejemplo de un evento de reintento de ejecución de un trabajo.

```
{
    "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"
    }
}
```

Etiquetado de recursos

Puede asignar sus propios metadatos a cada recurso mediante etiquetas que le ayuden a administrar sus recursos EMR sin servidor. En esta sección se proporciona una descripción general de las funciones de las etiquetas y se muestra cómo crearlas.

Temas

¿Qué es una etiqueta?

Una etiqueta es una etiqueta que se asigna a un AWS recurso. Cada etiqueta consta de una clave y un valor, ambos definidos por el usuario. Las etiquetas le permiten categorizar sus AWS los recursos por atributos, como el propósito, el propietario y el entorno. Cuando tenga muchos recursos del mismo tipo, puede identificar rápidamente un recurso específico en función de las etiquetas que le haya asignado. Por ejemplo, puede definir un conjunto de etiquetas para sus aplicaciones de Amazon EMR Serverless que le ayuden a realizar un seguimiento del propietario y el nivel de pila de cada aplicación. Le recomendamos que diseñe un conjunto coherente de claves de etiqueta para cada tipo de recurso.

Además, las etiquetas no se asignan a los recursos automáticamente. Después de añadir una etiqueta a un recurso, puede modificar el valor de una etiqueta o eliminarla del recurso en cualquier momento. Las etiquetas no tienen ningún significado semántico para Amazon EMR Serverless y se interpretan estrictamente como cadenas de caracteres. Si agrega una etiqueta con la misma clave que una etiqueta existente en ese recurso, el nuevo valor sobrescribirá al anterior.

Si las usaIAM, puede controlar qué usuarios de su AWS la cuenta tiene permiso para gestionar las etiquetas. Para ver ejemplos de políticas de control de acceso basadas en etiquetas, consulte Políticas para el control de acceso basado en etiquetas.

Etiquetado de recursos

Puede etiquetar aplicaciones y ejecuciones de tareas nuevas o existentes. Si utiliza Amazon EMR ServerlessAPI, el AWS CLI, o un AWS SDK, puede aplicar etiquetas a los nuevos recursos utilizando el `tags` parámetro de la API acción correspondiente. Puede aplicar etiquetas a los recursos existentes mediante la `TagResource` API acción.

Puede utilizar algunas acciones de creación de recursos para especificar etiquetas para un recurso al crear dicho recurso. En este caso, si las etiquetas no pueden aplicarse mientras se crea el recurso, este no podrá crearse. Este mecanismo garantiza que los recursos que pretendía etiquetar en el momento de su creación se creen con etiquetas específicas o no se creen en absoluto. Si etiqueta los recursos en el momento de la creación, no necesitará ejecutar scripts de etiquetado personalizados después de crear un recurso.

En la siguiente tabla se describen los recursos de Amazon EMR Serverless que se pueden etiquetar.

Recurso	Admite etiquetas	Admite la propagación de etiquetas	Admite el etiquetado en la creación (Amazon EMR ServerlessAPI, AWS CLI, y AWS SDK)	APIpara la creación (las etiquetas se pueden añadir durante la creación)
Aplicación	Sí	No. Las etiquetas asociadas a una aplicación no se propagan a las ejecuciones de trabajo enviadas a esa aplicación.	Sí	`CreateApplication`
Ejecución de trabajo	Sí	No	Sí	`StartJobRun`

Limitaciones de etiquetado

Las siguientes limitaciones básicas se aplican a las etiquetas:

- Cada recurso puede tener un máximo de 50 etiquetas creadas por el usuario.
- Para cada recurso, cada clave de etiqueta debe ser única y solo puede tener un valor.
- La longitud máxima de la clave es de 128 caracteres Unicode en UTF -8.
- La longitud máxima del valor es de 256 caracteres Unicode en UTF -8.
- Los caracteres permitidos son letras, números, espacios representables en UTF -8 y los siguientes caracteres: _.:/= + - @.
- Una clave de etiqueta no puede ser una cadena vacía. Un valor de etiqueta puede ser una cadena vacía, pero no nulo.
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.
- No utilice ningún prefijo `AWS:` ni ninguna combinación de mayúsculas o minúsculas como prefijo para las claves o los valores. Están reservadas únicamente para AWS usar.

Trabajar con etiquetas mediante el AWS CLI y Amazon EMR Serverless API

Usa lo siguiente AWS CLI comandos u API operaciones de Amazon EMR Serverless para añadir, actualizar, enumerar y eliminar las etiquetas de sus recursos.

Recurso	Admite etiquetas	Admite la propagación de etiquetas
Agregar o sobrescribir una o varias etiquetas.	`tag-resource`	`TagResource`
Enumera las etiquetas de un recurso	`list-tags-for-resource`	`ListTagsForResource`
Eliminar una o varias etiquetas .	`untag-resource`	`UntagResource`

En los siguientes ejemplos se muestra cómo etiquetar o desetiquetar los recursos mediante el AWS CLI.

Etiquete una aplicación existente

El siguiente comando etiqueta una aplicación existente.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

Quite la etiqueta de una aplicación existente

El siguiente comando elimina una etiqueta de una aplicación existente.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Listar las etiquetas de un recurso

El siguiente comando enumera las etiquetas asociadas a un recurso existente.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

Tutoriales para sistemas EMR sin servidor

En esta sección se describen los casos de uso más comunes cuando se trabaja con aplicaciones EMR sin servidor.

Temas

Uso de Java 17 con Amazon EMR Serverless

Con las EMR versiones 6.11.0 y posteriores de Amazon, puede configurar los trabajos de EMR Serverless Spark para que usen el tiempo de ejecución de Java 17 para la máquina virtual de Java (). JVM Utilice uno de los siguientes métodos para configurar Spark con Java 17.

`JAVA_HOME`

Para anular la JVM configuración de EMR Serverless 6.11.0 y versiones posteriores, puedes incluir la configuración en sus `spark.emr-serverless.driverEnv` clasificaciones y en las del `JAVA_HOME` entorno. `spark.executorEnv`

x86_64

Defina las propiedades necesarias para especificar Java 17 como `JAVA_HOME` configuración para el controlador y los ejecutores de 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 las propiedades necesarias para especificar Java 17 como JAVA_HOME configuración para el controlador y los ejecutores de 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, puedes especificar Java 17 en la `spark-defaults` clasificación para anular la JVM configuración de EMR Serverless 6.11.0 y versiones posteriores.

x86_64

Especifique Java 17 en la clasificación: `spark-defaults`

```
{
"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 en la `spark-defaults` clasificación:

```
{
```

```
"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/"
            }
         }
      ]
}
```

Uso de Apache Hudi con Serverless EMR

Para usar Apache Hudi con EMR aplicaciones sin servidor

1. Defina las propiedades de Spark requeridas en la ejecución de la tarea de Spark correspondiente.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Para sincronizar una tabla de Hudi con el catálogo configurado, designe una de las siguientes opciones AWS Glue Data Catalog como su metaalmacén o configure un metaalmacén externo. EMRServerless es compatible con las tablas de Hive para las cargas de trabajo de Hudi `hms` como modo de sincronización. EMRServerless activa esta propiedad por defecto. Para obtener más información sobre cómo configurar su metatienda, consulte. Configuración de Metastore

   > **Important**
   >
   > EMRServerless no admite `HIVEQL` ni ofrece opciones de modo de sincronización para `JDBC` que las tablas de Hive gestionen las cargas de trabajo de Hudi. Para obtener más información, consulta Modos de sincronización.

   Cuando usas el AWS Glue Data Catalog como metaalmacén, puede especificar las siguientes propiedades de configuración para su trabajo de 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 obtener más información sobre las versiones de Apache Hudi de AmazonEMR, consulta el historial de versiones de Hudi.

Uso de Apache Iceberg con Serverless EMR

Para usar Apache Iceberg con aplicaciones sin servidor EMR

1. Defina las propiedades de Spark requeridas en la ejecución de la tarea de Spark correspondiente.

   ```
   spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
   ```

2. Designe una de las siguientes opciones AWS Glue Data Catalog como su metaalmacén o configure un metaalmacén externo. Para obtener más información sobre cómo configurar su metaalmacén, consulte. Configuración de Metastore

   Configure las propiedades del metaalmacén que desee usar para Iceberg. Por ejemplo, si desea utilizar el AWS Glue Data Catalog, establezca las siguientes propiedades en la configuración de la aplicación.

   ```
   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
   ```

   Cuando utilice el AWS Glue Data Catalog como metaalmacén, puede especificar las siguientes propiedades de configuración para su trabajo de 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 obtener más información sobre las versiones de Apache Iceberg de AmazonEMR, consulta el historial de versiones de Iceberg.

Uso de bibliotecas de Python con EMR Serverless

Cuando ejecuta PySpark trabajos en aplicaciones Amazon EMR Serverless, puede empaquetar varias bibliotecas de Python como dependencias. Para ello, puede utilizar las funciones nativas de Python, crear un entorno virtual o configurar directamente sus PySpark trabajos para utilizar las bibliotecas de Python. En esta página se describe cada enfoque.

Uso de funciones nativas de Python

Si estableces la siguiente configuración, puedes utilizarla PySpark para cargar archivos de Python (`.py`), paquetes de Python comprimidos (`.zip`) y archivos Egg (`.egg`) a los ejecutores de Spark.

```
--conf spark.submit.pyFiles=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/<.py|.egg|.zip file>
```

Para obtener más información sobre cómo utilizar los entornos virtuales de Python para los PySpark trabajos, consulte Uso de funciones PySpark nativas.

Creación de un entorno virtual de Python

Para empaquetar varias bibliotecas de Python para un PySpark trabajo, puede crear entornos virtuales de Python aislados.

1. Para crear el entorno virtual de Python, utilice los siguientes comandos. El ejemplo que se muestra instala los paquetes `scipy` `matplotlib` en un paquete de entorno virtual y copia el archivo en una ubicación de Amazon S3.

   > **Important**
   >
   > Debe ejecutar los siguientes comandos en un entorno de Amazon Linux 2 similar con la misma versión de Python que utiliza en EMR Serverless, es decir, Python 3.7.10 para

Amazon EMR versión 6.6.0. Puede encontrar un ejemplo de Dockerfile en el repositorio Serverless Samples. EMR 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. Envía el trabajo de Spark con tus propiedades configuradas para usar el entorno virtual de 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
```

Tenga en cuenta que si no anula el binario de Python original, será `--conf spark.executorEnv.PYSPARK_PYTHON=python` la segunda configuración de la secuencia de ajustes anterior.

Para obtener más información sobre cómo utilizar los entornos virtuales de Python para los PySpark trabajos, consulte Uso de Virtualenv. Para ver más ejemplos de cómo enviar trabajos de Spark, consulta. Empleos en Spark

Configuración de PySpark trabajos para usar bibliotecas de Python

Con las EMR versiones 6.12.0 y posteriores de Amazon, puede configurar directamente los PySpark trabajos EMR sin servidor para que utilicen bibliotecas Python populares de ciencia de datos, como pandas NumPy, y PyArrowsin ninguna configuración adicional.

Los siguientes ejemplos muestran cómo empaquetar cada biblioteca de Python para un PySpark trabajo.

NumPy

NumPy es una biblioteca de Python para computación científica que ofrece matrices y operaciones multidimensionales para matemáticas, clasificación, simulación aleatoria y estadísticas básicas. Para utilizarla NumPy, ejecute el siguiente comando:

```
import numpy
```

pandas

pandas es una biblioteca de Python que se basa en. NumPy La biblioteca pandas proporciona a los científicos de datos estructuras de DataFramedatos y herramientas de análisis de datos. Para usar pandas, ejecuta el siguiente comando:

```
import pandas
```

PyArrow

PyArrow es una biblioteca de Python que administra datos en columnas en memoria para mejorar el rendimiento laboral. PyArrow se basa en la especificación de desarrollo multilenguaje Apache Arrow, que es una forma estándar de representar e intercambiar datos en formato de columnas. Para usarlo PyArrow, ejecute el siguiente comando:

```
import pyarrow
```

Uso de diferentes versiones de Python con EMR Serverless

Además del caso de usoUso de bibliotecas de Python con EMR Serverless, también puede utilizar entornos virtuales de Python para trabajar con versiones de Python diferentes a la versión incluida en la versión de Amazon EMR para su aplicación Amazon EMR Serverless. Para ello, debe crear un entorno virtual de Python con la versión de Python que desee utilizar.

Para enviar un trabajo desde un entorno virtual de Python

1. Cree su entorno virtual con los comandos del siguiente ejemplo. En este ejemplo, se instala Python 3.9.9 en un paquete de entorno virtual y se copia el archivo en una ubicación de Amazon S3.

> **Important**
>
> Si usa las EMR versiones 7.0.0 y posteriores de Amazon, debe ejecutar sus comandos en un entorno Amazon Linux 2023 similar al que usa para sus aplicaciones EMR sin servidor.
>
> Si utiliza la versión 6.15.0 o una versión anterior, debe ejecutar los siguientes comandos en un entorno de 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. Configura tus propiedades para usar el entorno virtual de Python y envía el trabajo de 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 obtener más información sobre cómo utilizar los entornos virtuales de Python para los PySpark trabajos, consulte Uso de Virtualenv. Para ver más ejemplos de cómo enviar trabajos de Spark, consulta. Empleos en Spark

Uso de Delta Lake OSS con EMR Serverless

Amazon, EMR versiones 6.9.0 y superiores

> **Note**
>
> Amazon EMR 7.0.0 y versiones posteriores utilizan Delta Lake 3.0.0, que cambia el nombre del `delta-core.jar` archivo a. `delta-spark.jar` Si utilizas Amazon EMR 7.0.0 o una versión superior, asegúrate de especificarlo `delta-spark.jar` en tus configuraciones.

Amazon EMR 6.9.0 y las versiones posteriores incluyen Delta Lake, por lo que ya no tendrá que empaquetar Delta Lake usted mismo ni indicar la `--packages` bandera con sus trabajos EMR sin servidor.

1. Cuando envíe trabajos EMR sin servidor, asegúrese de tener las siguientes propiedades de configuración e incluir los siguientes parámetros en el campo. `sparkSubmitParameters`

```
--conf spark.jars=/usr/share/aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/
delta-storage.jar
    --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension
    --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
```

2. Cree un local `delta_sample.py` para probar la creación y lectura de una tabla 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. Uso de AWS CLI, sube el `delta_sample.py` archivo a tu bucket de Amazon S3. A continuación, utilice el `start-job-run` comando para enviar un trabajo a una aplicación EMR sin servidor 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 de Python con Delta Lake, puede agregar la `delta-core` biblioteca empaquetándola como una dependencia o usándola como una imagen personalizada.

Como alternativa, puede utilizar `SparkContext.addPyFile` para añadir las bibliotecas de Python desde el `delta-core` JAR archivo:

```
import glob
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
spark.sparkContext.addPyFile(glob.glob("/usr/share/aws/delta/lib/delta-core_*.jar")[0])
```

Amazon, EMR versiones 6.8.0 y anteriores

Si utiliza Amazon EMR 6.8.0 o una versión anterior, siga estos pasos para usar Delta Lake OSS con sus aplicaciones EMR sin servidor.

1. Para crear una versión de código abierto de Delta Lake que sea compatible con la versión de Spark de su aplicación Amazon EMR Serverless, vaya a Delta GitHub y siga las instrucciones.
2. Cargue las bibliotecas de Delta Lake en un bucket de Amazon S3 de su Cuenta de AWS.
3. Cuando envíe trabajos EMR sin servidor en la configuración de la aplicación, incluya los JAR archivos de Delta Lake que se encuentran ahora en su depósito.

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
   ```

4. Para asegurarse de que puede leer y escribir desde una tabla Delta, ejecute una PySpark prueba de muestra.

   ```
   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
```

Envío de trabajos EMR sin servidor desde Airflow

El proveedor de Amazon en Apache Airflow proporciona operadores EMR sin servidor. Para obtener más información sobre los operadores, consulte Amazon EMR Serverless Operators en la documentación de Apache Airflow.

Se puede utilizar `EmrServerlessCreateApplicationOperator` para crear una aplicación Spark o Hive. También puedes utilizarla `EmrServerlessStartJobOperator` para iniciar uno o más trabajos con tu nueva aplicación.

Para usar el operador con Amazon Managed Workflows for Apache Airflow (MWAA) con Airflow 2.2.2, añade la siguiente línea a tu `requirements.txt` archivo y actualiza tu MWAA entorno para usar el nuevo archivo.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Tenga en cuenta que la compatibilidad con EMR Serverless se agregó a la versión 5.0.0 del proveedor Amazon. La versión 6.0.0 es la última versión compatible con Airflow 2.2.2. Puede utilizar versiones posteriores con Airflow 2.4.3 en adelante. MWAA

El siguiente ejemplo abreviado muestra cómo crear una aplicación, ejecutar varios trabajos de Spark y, a continuación, detener la aplicación. Hay un ejemplo completo disponible en el repositorio EMRServerless Samples GitHub . Para obtener detalles adicionales de la `sparkSubmit` configuración, consulteEmpleos en 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)
```

Uso de las funciones definidas por el usuario de Hive con Serverless EMR

Las funciones definidas por el usuario de Hive (UDFs) permiten crear funciones personalizadas para procesar registros o grupos de registros. En este tutorial, utilizará un ejemplo UDF con una aplicación Amazon EMR Serverless preexistente para ejecutar un trabajo que genere un resultado de consulta. Para obtener información sobre cómo configurar una aplicación, consulte. Introducción a Amazon EMR Serverless

Para usar un UDF con EMR Serverless

1. Navegue hasta el GitHubpara ver una muestraUDF. Clona el repositorio y cambia a la rama de git que quieras usar. Actualiza `maven-compiler-plugin` el `pom.xml` archivo del repositorio para tener una fuente. Actualice también la configuración de la versión java de destino `a1.8`. Ejecute `mvn package -DskipTests` para crear el JAR archivo que contiene la muestraUDFs.
2. Tras crear el JAR archivo, cárguelo en su bucket de S3 con el siguiente comando.

```
aws s3 cp brickhouse-0.8.2-JS.jar s3://DOC-EXAMPLE-BUCKET/jars/
```

3. Cree un archivo de ejemplo para usar una de las UDF funciones de ejemplo. Guarde esta consulta como `udf_example.q` y cárguela en su bucket de 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. Envía el siguiente trabajo de 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. Utilice el `get-job-run` comando para comprobar el estado de su trabajo. Espera a que el estado cambie a`SUCCESS`.

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

6. Descargue los archivos de salida con el siguiente comando.

```
aws s3 cp --recursive s3://DOC-EXAMPLE-BUCKET/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

El `stdout.gz` archivo es similar al siguiente.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

Uso de imágenes personalizadas con EMR Serverless

Temas

- Usa una versión personalizada de Python
- Usa una versión Java personalizada
- Crea una imagen de ciencia de datos
- Procesamiento de datos geoespaciales con Apache Sedona

Usa una versión personalizada de Python

Puede crear una imagen personalizada para usar una versión diferente de Python. Para usar la versión 3.10 de Python para los trabajos de Spark, por ejemplo, ejecuta el siguiente 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 el trabajo de Spark, configura tus propiedades para usar el entorno virtual de Python, de la siguiente manera.

```
--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
```

Usa una versión Java personalizada

El siguiente ejemplo muestra cómo crear una imagen personalizada para usar Java 11 en tus trabajos de 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 el trabajo de Spark, configura las propiedades de Spark para que usen Java 11, de la siguiente manera.

```
--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-
```

Crea una imagen de ciencia de datos

El siguiente ejemplo muestra cómo incluir paquetes Python comunes de ciencia de datos, como Pandas y 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
```

Procesamiento de datos geoespaciales con Apache Sedona

El siguiente ejemplo muestra cómo crear una imagen para incluir Apache Sedona para el procesamiento 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
```

Uso de la integración de Amazon Redshift para Apache Spark en Amazon Serverless EMR

Con la EMR versión 6.9.0 y posteriores de Amazon, cada imagen de la versión incluye un conector entre Apache Spark y Amazon Redshift. Con este conector, puede usar Spark en Amazon EMR Serverless para procesar los datos almacenados en Amazon Redshift. La integración se basa en el conector de código abierto `spark-redshift`. En el caso de Amazon EMR Serverless, la integración de Amazon Redshift para Apache Spark se incluye como integración nativa.

Temas

Lanzamiento de una aplicación Spark con la integración de Amazon Redshift para Apache Spark

Para usar la integración con EMR Serverless 6.9.0, debes pasar las dependencias de Spark-Redshift requeridas con tu trabajo de Spark. Se utiliza `--jars` para incluir bibliotecas relacionadas con el conector Redshift. Para ver otras ubicaciones de archivos compatibles con la opción `--jars`, consulte la sección Administración avanzada de dependencias de la documentación de Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

EMRLas versiones 6.10.0 y posteriores de Amazon no requieren la `minimal-json.jar` dependencia e instalan automáticamente las demás dependencias en cada clúster de forma predeterminada. En los siguientes ejemplos se muestra cómo lanzar una aplicación de Spark con la integración de Amazon Redshift para Apache Spark.

Amazon EMR 6.10.0 +

Lance un trabajo de Spark en Amazon EMR Serverless con la integración de Amazon Redshift para Apache Spark EMR on Serverless, versión 6.10.0 y versiones posteriores.

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

Para lanzar un trabajo de Spark en Amazon EMR Serverless con la integración de Amazon Redshift para Apache Spark EMR en la versión 6.9.0 de Serverless, utilice `--jars` la opción que

se muestra en el siguiente ejemplo. Tenga en cuenta que las rutas que aparecen en la lista de la `--jars` opción son las rutas predeterminadas de los archivos. JAR

```
--jars
    /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
    /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar
```

```
spark-submit \
  --jars /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,/usr/share/aws/redshift/
spark-redshift/lib/spark-redshift.jar,/usr/share/aws/redshift/spark-redshift/lib/
spark-avro.jar,/usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar \
  my_script.py
```

Autenticación con la integración de Amazon Redshift para Apache Spark

Uso AWS Secrets Manager para recuperar las credenciales y conectarse a Amazon Redshift

Puede autenticarse de forma segura en Amazon Redshift almacenando las credenciales en Secrets Manager y haciendo que el trabajo de Spark llame al para `GetSecretValue` API buscarlas:

```
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
```

Autenticarse en Amazon Redshift con un controlador JDBC

Establezca el nombre de usuario y la contraseña dentro del JDBC URL

Puede autenticar un trabajo de Spark en un clúster de Amazon Redshift especificando el nombre y la contraseña de la base de datos de Amazon Redshift en. JDBC URL

> **Note**
>
> Si pasa las credenciales de la base de datosURL, cualquier persona que tenga acceso a ellas también URL podrá acceder a las credenciales. Por lo general, no se recomienda este método porque no es seguro.

Si la seguridad no es un problema para su aplicación, puede utilizar el siguiente formato para establecer el nombre de usuario y la contraseña en JDBCURL:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Autenticación IAM basada en el uso con la función de ejecución de trabajos de Amazon EMR Serverless

A partir de la versión 6.9.0 de Amazon EMR Serverless, el controlador Amazon JDBC Redshift 2.1 o superior se incluye en el entorno. Con JDBC el controlador 2.1 y versiones posteriores, puede especificar el nombre de usuario JDBC URL y la contraseña sin procesar, pero no incluirlos.

En su lugar, puede especificar un esquema `jdbc:redshift:iam://`. Esto le ordena al JDBC controlador que utilice su función de ejecución de trabajos EMR sin servidor para obtener las credenciales automáticamente. Consulte Configurar una ODBC conexión JDBC o una conexión para usar IAM credenciales en la Guía de administración de Amazon Redshift para obtener más información. Un ejemplo de ello URL es:

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Los siguientes permisos son necesarios para su función de ejecución de tareas cuando se cumplen las condiciones establecidas:

Permiso	Condiciones en las que se requiere un rol de ejecución de trabajos
`redshift:GetClusterCredentials`	Necesario para que el JDBC conductor obtenga las credenciales de Amazon Redshift
`redshift:DescribeCluster`	Obligatorio si especifica el clúster de Amazon Redshift y Región de AWS en JDBC URL lugar de en el punto final
`redshift-serverless:GetCredentials`	Necesario para que el JDBC conductor obtenga las credenciales de Amazon Redshift Serverless
`redshift-serverless:GetWorkgroup`	Obligatorio si utiliza Amazon Redshift Serverless y especifica el nombre del URL grupo de trabajo y la región

Conexión a Amazon Redshift desde un sitio diferente VPC

Al configurar un clúster de Amazon Redshift aprovisionado o un grupo de trabajo Amazon Redshift Serverless en VPC una, debe configurar la VPC conectividad de la aplicación EMR Amazon Serverless para acceder a los recursos. Para obtener más información sobre cómo configurar la VPC conectividad en una aplicación sin servidor, consulte. EMR Configurar VPC el acceso

- Si el clúster de Amazon Redshift aprovisionado o el grupo de trabajo Amazon Redshift Serverless son de acceso público, puede especificar una o más subredes privadas que tengan NAT una puerta de enlace adjunta al crear aplicaciones sin servidor. EMR
- Si el clúster de Amazon Redshift o el grupo de trabajo sin servidor de Amazon Redshift aprovisionados no son de acceso público, debe crear un VPC punto de conexión gestionado de Amazon Redshift para su clúster de Amazon Redshift, tal y como se describe en. Configurar VPC el acceso Como alternativa, puede crear su grupo de trabajo Amazon Redshift Serverless tal y como se describe en Conexión a Amazon Redshift Serverless de la Guía de administración de Amazon Redshift. Debe asociar el clúster o el subgrupo a las subredes privadas que especifique al crear la aplicación sin servidor. EMR

> **Note**
>
> Si utiliza la autenticación IAM basada y las subredes privadas de la aplicación EMR sin servidor no tienen una NAT puerta de enlace adjunta, también debe crear un VPC punto de conexión en esas subredes para Amazon Redshift o Amazon Redshift Serverless. De esta forma, el conductor puede obtener las credencialesJDBC.

Lectura y escritura desde y hacia Amazon Redshift

Los siguientes ejemplos de código se utilizan PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con una fuente de datos API y con Spark. SQL

Data source API

Se utiliza PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con fuente de datos. 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

Se utiliza PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con 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()
```

Consideraciones y limitaciones al utilizar el conector de Spark

- Te recomendamos que actives la JDBC conexión de Spark en Amazon EMR a Amazon Redshift. SSL
- Le recomendamos que administre las credenciales del clúster de Amazon Redshift en AWS Secrets Manager como práctica recomendada. Consulte Uso AWS Secrets Manager para recuperar las credenciales para conectarse a Amazon Redshift, por ejemplo.
- Le recomendamos que asigne un IAM rol con el parámetro `aws_iam_role` de autenticación de Amazon Redshift.
- Actualmente, el parámetro `tempformat` no admite el formato Parquet.
- Los `tempdir` URI puntos apuntan a una ubicación de Amazon S3. Este directorio temporal no se limpia automáticamente y, por lo tanto, podría agregar costos adicionales.
- Tenga en cuenta las siguientes recomendaciones para Amazon Redshift:
  - Le recomendamos que bloquee el acceso público al clúster de Amazon Redshift.
  - Le recomendamos que active el registro de auditoría de Amazon Redshift.
  - Le recomendamos que active el cifrado en reposo de Amazon Redshift.
- Tenga en cuenta las siguientes recomendaciones para Amazon S3:
  - Le recomendamos que bloquee el acceso público a los buckets de Amazon S3.
  - Le recomendamos que utilice el cifrado del servidor de Amazon S3 para cifrar los buckets de Amazon S3 utilizados.
  - Le recomendamos que utilice las políticas de ciclo de vida de Amazon S3 para definir las reglas de retención del bucket de Amazon S3.
  - Amazon EMR siempre verifica el código importado de código abierto a la imagen. Por motivos de seguridad, no admitimos los siguientes métodos de autenticación de Spark a Amazon S3:
    - Opción AWS claves de acceso en la clasificación de configuración `hadoop-env`
    - Codificación AWS claves de acceso en el `tempdir` URI

Para obtener más información sobre el uso del conector y sus parámetros compatibles, consulte los siguientes recursos:

- Integración de Amazon Redshift para Apache Spark en la Guía de administración de Amazon Redshift
- Repositorio comunitario de `spark-redshift` en GitHub

Conexión a DynamoDB con Amazon Serverless EMR

En este tutorial, cargará un subconjunto de datos de la Junta de Nombres Geográficos de los Estados Unidos a un bucket de Amazon S3 y, a continuación, utilizará Hive o Spark en Amazon EMR Serverless para copiar los datos en una tabla de Amazon DynamoDB que pueda consultar.

Paso 1: Cargar datos a un bucket de Amazon S3

Para crear un depósito de Amazon S3, siga las instrucciones de la Guía del usuario de la consola de Amazon Simple Storage Service Console. Sustituya las referencias a *DOC-EXAMPLE-BUCKET* por el nombre del depósito recién creado. Ahora su aplicación EMR sin servidor está lista para ejecutar tareas.

1. Descargue el archivo de datos de muestra `features.zip` con el siguiente comando.

```
wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
features.zip
```

2. Extraiga el `features.txt` archivo del archivo y visualice las primeras líneas del archivo:

```
unzip features.zip
head features.txt
```

El resultado debería tener un aspecto similar al siguiente.

```
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
```

Los campos de cada línea indican un identificador único, un nombre, un tipo de elemento natural, un estado, una latitud en grados, una longitud en grados y una altura en pies.

3. Cargue sus datos a Amazon S3

```
aws s3 cp features.txt s3://DOC-EXAMPLE-BUCKET/features/
```

Paso 2: Crear una tabla de Hive

Utilice Apache Spark o Hive para crear una nueva tabla de Hive que contenga los datos cargados en Amazon S3.

Spark

Para crear una tabla de Hive con Spark, ejecute el siguiente comando.

```
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';")
```

Ahora tienes una tabla Hive rellena con los datos del `features.txt` archivo. Para comprobar que tus datos están en la tabla, ejecuta una SQL consulta de Spark como se muestra en el siguiente ejemplo.

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Para crear una tabla de Hive con Hive, ejecuta el siguiente comando.

```
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';
```

Ahora tiene una tabla Hive que contiene los datos del archivo. `features.txt` Para comprobar que los datos están en la tabla, ejecute una consulta de HiveQL, como se muestra en el siguiente ejemplo.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

Paso 3: Copiar datos a DynamoDB

Use Spark o Hive para copiar los datos a una nueva tabla de DynamoDB.

Spark

Para copiar datos de la tabla Hive que creó en el paso anterior a DynamoDB, siga los pasos 1 a 3 de Copiar datos a DynamoDB. Esto crea una nueva tabla de DynamoDB llamada. `Features` A continuación, puede leer los datos directamente del archivo de texto y copiarlos en la tabla de DynamoDB, como se muestra en el siguiente ejemplo.

```
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 datos de la tabla Hive que creó en el paso anterior a DynamoDB, siga las instrucciones de Copiar datos a DynamoDB.

Paso 4: Consulta de datos de DynamoDB

Utilice Spark o Hive para consultar la tabla de DynamoDB.

Spark

Para consultar datos de la tabla de DynamoDB que creó en el paso anterior, puede usar SQL Spark o Spark. MapReduce API

Example — Consulta tu tabla de DynamoDB con Spark SQL

La siguiente SQL consulta de Spark devuelve una lista de todos los tipos de funciones en orden alfabético.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

La siguiente SQL consulta de Spark devuelve una lista de todos los lagos que comienzan por la 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;")
```

La siguiente SQL consulta de Spark devuelve una lista de todos los estados con al menos tres entidades que se encuentran a más de una milla.

```
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 — Consulta tu tabla de DynamoDB con el Spark MapReduce API

La siguiente MapReduce consulta devuelve una lista de todos los tipos de funciones en orden alfabético.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .map(pair => pair._2.get("feature_class").getS)
    .distinct()
    .sortBy(value => value)
    .toDF("feature_class")
```

La siguiente MapReduce consulta devuelve una lista de todos los lagos que comienzan por la 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")
```

La siguiente MapReduce consulta devuelve una lista de todos los estados con al menos tres entidades que se encuentran a más de una milla.

```
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 datos de la tabla de DynamoDB que creó en el paso anterior, siga las instrucciones de Consulta los datos de la tabla de DynamoDB.

Configuración del acceso entre cuentas

Para configurar el acceso multicuenta para EMR Serverless, complete los siguientes pasos. En el ejemplo, `AccountA` es la cuenta en la que creó la aplicación Amazon EMR Serverless y `AccountB` es la cuenta en la que se encuentra su Amazon DynamoDB.

1. Cree una tabla de DynamoDB en. `AccountB` Para obtener más información, consulte el paso 1: Crear una tabla.
2. Cree un `Cross-Account-Role-B` IAM rol `AccountB` que pueda acceder a la tabla de DynamoDB.

   a. Inicie sesión en AWS Management Console y abre la IAM consola en https://console.aws.amazon.com/iam/.

   b. Elija Roles y cree un nuevo rol llamado`Cross-Account-Role-B`. Para obtener más información sobre cómo crear IAM roles, consulte Creación de IAM roles en la guía del usuario.

   c. Cree una IAM política que conceda permisos para acceder a la tabla multicuenta de DynamoDB. A continuación, adjunte la política aIAM. `Cross-Account-Role-B`

      La siguiente es una política que concede acceso a una tabla de DynamoDB`CrossAccountTable`.

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

   d. Edite la relación de confianza del rol `Cross-Account-Role-B`.

      Para configurar la relación de confianza del rol, seleccione la pestaña Relaciones de confianza de la IAM consola para el rol que creó en el paso 2: Cross-Account-Role-B.

Seleccione Editar relación de confianza y, a continuación, añada el siguiente documento de política. Este documento permite `Job-Execution-Role-A` `AccountA` a Int asumir esta `Cross-Account-Role-B` función.

```
{"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 - STS Assume role` permisos para asumir`Cross-Account-Role-B`.

En la IAM consola para Cuenta de AWS `AccountA`, seleccione`Job-Execution-Role-A`. Agregue la siguiente instrucción de política al `Job-Execution-Role-A` para denegar la acción `AssumeRole` en el rol `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 la `dynamodb.customAWSCredentialsProvider` propiedad con un valor como `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` en la clasificación de sitios principales. Establezca la variable de entorno `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` con el ARN valor de. `Cross-Account-Role-B`

3. Ejecute el trabajo de Spark o Hive usando`Job-Execution-Role-A`.

Consideraciones

Consideraciones al usar el conector de DynamoDB con Apache Spark

- Spark SQL no admite la creación de una tabla Hive con la opción de gestor de almacenamiento. Para obtener más información, consulte Especificar el formato de almacenamiento para las tablas de Hive en la documentación de Apache Spark.
- Spark SQL no admite la `STORED BY` operación con el controlador de almacenamiento. Si desea interactuar con una tabla de DynamoDB a través de una tabla Hive externa, utilice Hive para crear primero la tabla.
- Para convertir una consulta en una consulta de DynamoDB, el conector de DynamoDB utiliza el comando push down de predicados. La pulsación de predicados filtra los datos por una columna que está asignada a la clave de partición de una tabla de DynamoDB. La pulsación de predicados solo funciona cuando se utiliza el conector con Spark y no con el. SQL MapReduce API

Consideraciones al utilizar el conector de DynamoDB con Apache Hive

Ajustar el número máximo de mapeadores

- Si utiliza la `SELECT` consulta para leer datos de una tabla Hive externa que se asigna a DynamoDB, el número de tareas de mapa EMR en Serverless se calcula como el rendimiento de lectura total configurado para la tabla de DynamoDB, dividido por el rendimiento por tarea de mapa. El rendimiento predeterminado por tarea de mapa es 100.
- El trabajo de Hive puede utilizar un número de tareas de mapa superior al número máximo de contenedores configurados por aplicación EMR sin servidor, en función del rendimiento de lectura configurado para DynamoDB. Además, una consulta de Hive de ejecución prolongada puede consumir toda la capacidad de lectura aprovisionada de la tabla de DynamoDB. Esto afecta negativamente a otros usuarios.
- Puede usar la `dynamodb.max.map.tasks` propiedad para establecer un límite superior para las tareas de mapas. También puede usar esta propiedad para ajustar la cantidad de datos que lee cada tarea de mapa en función del tamaño del contenedor de tareas.
- Puede establecer la `dynamodb.max.map.tasks` propiedad en el nivel de consulta de Hive o en la `hive-site` clasificación del start-job-run comando. Este valor debe ser igual o superior a 1. Cuando Hive procesa la consulta, el trabajo de Hive resultante no utiliza más que los valores de `dynamodb.max.map.tasks` cuando lee de la tabla de DynamoDB.

Ajustar el rendimiento de escritura por tarea

- El rendimiento de escritura por tarea en EMR Serverless se calcula dividiendo el rendimiento de escritura total configurado para una tabla de DynamoDB por el valor de la propiedad. `mapreduce.job.maps` En Hive, el valor predeterminado de esta propiedad es 2. Por lo tanto, las dos primeras tareas de la fase final del trabajo de Hive pueden consumir todo el rendimiento de escritura. Esto lleva a limitar la escritura de otras tareas del mismo trabajo o de otros trabajos.
- Para evitar la limitación de la escritura, puede establecer el valor de la `mapreduce.job.maps` propiedad en función del número de tareas de la etapa final o del rendimiento de escritura que desee asignar por tarea. Establezca esta propiedad en la `mapred-site` clasificación del start-job-run comando en Serverless. EMR

Seguridad

Seguridad en la nube en AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS y tú. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que se ejecuta AWS servicios en el AWS Nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte del AWS programas de cumplimiento. Para obtener más información sobre los programas de conformidad que se aplican a Amazon EMR Serverless, consulte AWS servicios incluidos en el ámbito de aplicación por programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por la AWS servicio que utiliza. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos vigentes.

Esta documentación le ayuda a entender cómo aplicar el modelo de responsabilidad compartida al utilizar Amazon EMR Serverless. Los temas de este capítulo muestran cómo configurar Amazon EMR Serverless y utilizar otros AWS servicios para cumplir sus objetivos de seguridad y conformidad.

Temas

- Resiliencia en Amazon EMR Serverless
- Seguridad de la infraestructura en Amazon EMR Serverless
- Análisis de configuración y vulnerabilidad en Amazon EMR Serverless

Prácticas recomendadas de seguridad para Amazon EMR Serverless

Amazon EMR Serverless proporciona una serie de características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no constituyen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

Aplicación del principio de privilegios mínimos

EMRServerless proporciona una política de acceso granular para las aplicaciones que utilizan IAM funciones, como las de ejecución. Recomendamos que a los roles de ejecución solo se les otorguen los privilegios mínimos necesarios para el trabajo, como cubrir su aplicación y el acceso al destino del registro. También recomendamos auditar los trabajos para detectar permisos de forma regular y ante cualquier cambio en el código de la aplicación.

Aislar el código de aplicación no confiable

EMRServerless crea un aislamiento total de la red entre los trabajos que pertenecen a diferentes aplicaciones EMR sin servidor. En los casos en los que se desee aislar los trabajos a nivel de trabajo, considere la posibilidad de aislar los trabajos en diferentes aplicaciones sin servidor. EMR

Control de acceso basado en roles () permisos RBAC

Los administradores deben controlar estrictamente los permisos del control de acceso basado en roles para las aplicaciones RBAC sin servidor. EMR

Protección de datos

La AWS el modelo de responsabilidad compartida se aplica a la protección de datos en Amazon EMR Serverless. Como se describe en este modelo, AWS es responsable de proteger la

infraestructura global en la que se ejecutan todos los AWS Nube. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Este contenido incluye las tareas de configuración y administración de la seguridad del AWS servicios que utiliza. Para obtener más información sobre la privacidad de los datos, consulte la sección Privacidad de datos FAQ. Para obtener información sobre la protección de datos en Europa, consulte la AWS Modelo de responsabilidad compartida y entrada de GDPR blog sobre AWS Blog de seguridad.

Para fines de protección de datos, le recomendamos que proteja AWS credenciales de la cuenta y configurar cuentas individuales con AWS Identity and Access Management (IAM). De esta manera, cada usuario recibe únicamente los permisos necesarios para cumplir con sus obligaciones laborales. También recomendamos proteger sus datos de las siguientes maneras:

- Utilice la autenticación multifactorial (MFA) con cada cuenta.
- UtiliceSSL/TLSpara comunicarse con AWS recursos. Recomendamos la TLS versión 1.2 o posterior.
- Configure API y registre la actividad de los usuarios con AWS CloudTrail.
- Uso AWS soluciones de cifrado, junto con todos los controles de seguridad predeterminados AWS servicios.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.
- Utilice las opciones de cifrado de Amazon EMR Serverless para cifrar los datos en reposo y en tránsito.
- Si necesita entre FIPS 140 y 2 módulos criptográficos validados para acceder AWS a través de una interfaz de línea de comandos oAPI, utilice un FIPS punto final. Para obtener más información sobre los FIPS puntos finales disponibles, consulte la Norma Federal de Procesamiento de Información (FIPS) 140-2.

Le recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, números de cuenta de sus clientes, en los campos de formato libre, como el campo Nombre. Esto incluye cuando trabaja con Amazon EMR Serverless u otro AWS servicios que utilizan la consola, API AWS CLI, o AWS SDKs. Es posible que cualquier dato que introduzca en Amazon EMR Serverless u otros servicios se recoja para incluirlo en los registros de diagnóstico. Cuando proporcione un mensaje URL a un servidor externo, no incluya información sobre las credenciales en él URL para validar su solicitud a ese servidor.

Cifrado en reposo

El cifrado de datos ayuda a impedir que los usuarios no autorizados lean los datos en un clúster y sistemas de almacenamiento de datos asociados. Esto incluye los datos guardados en medios persistentes, conocidos como datos en reposo y datos que pueden ser interceptados cuando recorren la red, conocidos como datos en tránsito.

El cifrado de datos requiere las claves y los certificados. Puedes elegir entre varias opciones, incluidas las claves administradas por AWS Key Management Service, claves gestionadas por Amazon S3 y claves y certificados de proveedores personalizados que usted suministre. Cuando se utiliza AWS KMS como proveedor de claves, se aplican cargos por el almacenamiento y el uso de las claves de cifrado. Para obtener más información, consulte AWS KMS precios.

Antes de especificar las opciones de cifrado, decida qué sistemas de administración de claves y certificados quiere usar. A continuación, cree las claves y los certificados para los proveedores personalizados que especifique como parte de la configuración de cifrado.

Cifrado en reposo para EMRFS datos en Amazon S3

Cada aplicación EMR sin servidor utiliza una versión de lanzamiento específica, que incluye EMRFS (Sistema de EMR archivos). El cifrado de Amazon S3 funciona con los objetos del Sistema de EMR Archivos (EMRFS) leídos y escritos en Amazon S3. Puede especificar el cifrado del lado del servidor (SSE) o el cifrado del lado del cliente () de Amazon S3 como modo de cifrado predeterminado cuando habilita el cifrado en reposo. CSE También puede especificar métodos de cifrado diferentes para buckets individuales utilizando Per bucket encryption overrides (Reemplazos de cifrado por bucket). Independientemente de si el cifrado de Amazon S3 está activado, Transport Layer Security (TLS) cifra los EMRFS objetos en tránsito entre los nodos EMR del clúster y Amazon S3. Si utiliza Amazon S3 CSE con claves administradas por el cliente, el rol de ejecución que utilice para ejecutar trabajos en una aplicación EMR sin servidor debe tener acceso a la clave. Para obtener información detallada sobre el cifrado de Amazon S3, consulte Protección de datos mediante cifrado en la Guía para desarrolladores de Amazon Simple Storage Service.

> **Note**
>
> Cuando usas AWS KMS, se aplican cargos por el almacenamiento y el uso de las claves de cifrado. Para obtener más información, consulte AWS KMS precios.

Cifrado del servidor de Amazon S3

Cuando configura el cifrado del servidor de Amazon S3, Amazon S3 cifra los datos del objeto a medida que escribe los datos en el disco y descifra los datos cuando se accede. Para obtener más informaciónSSE, consulte Protección de datos mediante cifrado del lado del servidor en la Guía para desarrolladores de Amazon Simple Storage Service.

Puede elegir entre dos sistemas de administración de claves diferentes cuando lo especifique SSE en Amazon EMR Serverless:

- SSE-S3 - Amazon S3 administra las claves por usted. No se requiere ninguna configuración adicional en EMR Serverless.
- SSE- KMS - Utilizas un AWS KMS key para configurarlo con políticas adecuadas para EMR Serverless. No se requiere ninguna configuración adicional en EMR Serverless.

Para utilizar AWS KMS para el cifrado de los datos que escribe en Amazon S3, tiene dos opciones cuando utiliza el `StartJobRun`API. Puede habilitar el cifrado de todo lo que escriba en Amazon S3 o puede habilitar el cifrado de los datos que escriba en un bucket específico. Para obtener más información al respecto `StartJobRun`API, consulte la Referencia sobre sistemas EMRsin servidor API.

Para activarlo AWS KMS para cifrar todos los datos que escriba en Amazon S3, utilice los siguientes comandos cuando llame al `StartJobRun`API.

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Para activar AWS KMS para cifrar los datos que escribas en un depósito específico, utiliza los siguientes comandos cuando llames al `StartJobRun`API.

```
--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>
```

SSEcon claves proporcionadas por el cliente (SSE-C) no está disponible para su uso con Serverless. EMR

Cifrado del cliente de Amazon S3

Con el cifrado del lado del cliente de Amazon S3, el cifrado y el descifrado de Amazon S3 se llevan a cabo en el EMRFS cliente disponible en todas las versiones de Amazon. EMR Los objetos se cifran antes de cargarlos en Amazon S3 y se descifran después de que se descarguen. El proveedor que especifique proporciona la clave de cifrado que utiliza el cliente. El cliente puede utilizar las claves proporcionadas por AWS KMS (CSE-KMS) o una clase Java personalizada que proporcione la clave raíz del lado del cliente (CSE-C). Las especificaciones de cifrado varían ligeramente entre CSE - KMS y CSE -C, según el proveedor especificado y los metadatos del objeto que se va a descifrar o cifrar. Si utiliza Amazon S3 CSE con claves administradas por el cliente, el rol de ejecución que utilice para ejecutar trabajos en una aplicación EMR sin servidor debe tener acceso a la clave. Se pueden aplicar KMS cargos adicionales. Para obtener más información sobre estas diferencias, consulte Protección de datos mediante el cifrado del lado del cliente en la Guía para desarrolladores de Amazon Simple Storage Service.

Cifrado de disco local

Los datos almacenados en un almacenamiento efímero se cifran con claves propias del servicio mediante el algoritmo criptográfico -256 estándar AES del sector.

Administración de claves

Puede configurarlo para que rote automáticamente KMS las claves. KMS De este modo, las claves se rotan una vez al año y se guardan las antiguas de forma indefinida para poder seguir descifrando los datos. Para obtener información adicional, consulte Rotación de las llaves maestras del cliente.

Cifrado en tránsito

Las siguientes funciones de cifrado específicas de la aplicación están disponibles en Amazon EMR Serverless:

- Spark
  - De forma predeterminada, la comunicación entre los controladores y los ejecutores de Spark está autenticada y es interna. RPCla comunicación entre los controladores y los ejecutores está cifrada.
- Hive
  - La comunicación entre AWS Glue Metastore y las aplicaciones EMR Serverless se realizan a través de. TLS

Solo debe permitir las conexiones cifradas a través de HTTPS (TLS) mediante la SecureTransport condición aws: de IAM las políticas de bucket de Amazon S3.

Identity and Access Management (IAM) en Amazon EMR Serverless

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS recursos. IAMlos administradores controlan quién puede autenticarse (iniciar sesión) y quién está autorizado (tiene permisos) para usar los recursos de Amazon EMR Serverless. IAMes un Servicio de AWS que puede utilizar sin coste adicional.

Temas

Público

Cómo se usa AWS Identity and Access Management (IAM) varía según el trabajo que realice en Amazon EMR Serverless.

Usuario del servicio: si utiliza el servicio Amazon EMR Serverless para realizar su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más funciones de Amazon EMR Serverless para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una función de Amazon EMR Serverless, consulteSolución de problemas de identidad y acceso a Amazon EMR Serverless.

Administrador de servicios: si está a cargo de los recursos de Amazon EMR Serverless en su empresa, probablemente tenga acceso total a Amazon EMR Serverless. Es su trabajo determinar a qué funciones y recursos de Amazon EMR Serverless deben acceder los usuarios del servicio. A continuación, debe enviar solicitudes a su IAM administrador para cambiar los permisos de los usuarios del servicio. Revise la información de esta página para comprender los conceptos básicos deIAM. Para obtener más información sobre cómo su empresa puede utilizar IAM Amazon EMR Serverless, consulteIdentity and Access Management (IAM) en Amazon EMR Serverless.

IAMadministrador: si es IAM administrador, puede que le interese obtener más información sobre cómo redactar políticas para administrar el acceso a Amazon EMR Serverless. Para ver ejemplos de políticas basadas en la identidad de Amazon EMR Serverless que puede utilizar, consulte. IAM Ejemplos de políticas basadas en la identidad para Serverless EMR

Autenticación con identidades

La autenticación es la forma de iniciar sesión en AWS utilizando tus credenciales de identidad. Debe estar autenticado (haber iniciado sesión en AWS) como Usuario raíz de la cuenta de AWS, como IAM usuario o asumiendo un IAM rol.

Puede iniciar sesión en AWS como identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios de (IAMIdentity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, el administrador configuró previamente la federación de identidades mediante roles. IAM Cuando accedes AWS al usar la federación, está asumiendo un rol de manera indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en AWS Management Console o el AWS portal de acceso. Para obtener más información sobre cómo iniciar sesión en AWS, consulta Cómo iniciar sesión en tu Cuenta de AWS en la AWS Sign-In Guía del usuario.

Si accedes AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no usa AWS herramientas, debe firmar las solicitudes usted mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar AWS APIsolicitudes en la Guía IAM del usuario.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo: AWS recomienda que utilice la autenticación multifactorial (MFA) para aumentar la seguridad de su cuenta. Para obtener más información,

consulte Autenticación multifactorial en la AWS IAM Identity Center Guía del usuario y Uso de la autenticación multifactorial () MFA en AWS en la Guía del usuario de IAM.

Cuenta de AWS usuario raíz

Al crear un Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todas Servicios de AWS y los recursos de la cuenta. Esta identidad se denomina Cuenta de AWS usuario root y se accede a él iniciando sesión con la dirección de correo electrónico y la contraseña que utilizó para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de tareas que requieren que inicie sesión como usuario root, consulte Tareas que requieren credenciales de usuario root en la Guía del IAM usuario.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web, el AWS Directory Service, el directorio del Centro de identidades o cualquier usuario que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para la administración centralizada del acceso, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en IAM Identity Center, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todos sus Cuentas de AWS y aplicaciones. Para obtener información sobre IAM Identity Center, consulte ¿Qué es IAM Identity Center? en el AWS IAM Identity Center Guía del usuario.

Usuarios y grupos de IAM

Un IAMusuario es una identidad dentro de tu Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos utilizar credenciales temporales en lugar de crear IAM usuarios con credenciales de larga duración, como contraseñas y claves de acceso. Sin embargo, si tiene casos de uso específicos que requieren credenciales a largo plazo con IAM los usuarios, le recomendamos que rote las claves de acceso. Para obtener

más información, consulte Rotar las claves de acceso con regularidad para los casos de uso que requieran credenciales de larga duración en la Guía del IAM usuario.

Un IAMgrupo es una identidad que especifica un conjunto de IAM usuarios. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar IAM los recursos.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Cuándo crear un IAM usuario (en lugar de un rol) en la Guía del IAM usuario.

IAMroles

Un IAMrol es una identidad dentro de tu Cuenta de AWS que tiene permisos específicos. Es similar a un IAM usuario, pero no está asociado a una persona específica. Puede asumir temporalmente un IAM rol en el AWS Management Console cambiando de rol. Puede asumir un rol llamando a un AWS CLI o AWS APIoperación o mediante una operación personalizadaURL. Para obtener más información sobre los métodos de uso de roles, consulte Uso de IAM roles en la Guía del IAM usuario.

IAMlos roles con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información sobre los roles para la federación, consulte Creación de un rol para un proveedor de identidad externo en la Guía del IAM usuario. Si usa IAM Identity Center, configura un conjunto de permisos. Para controlar a qué pueden acceder sus identidades después de autenticarse, IAM Identity Center correlaciona el conjunto de permisos con un rol en. IAM Para obtener información sobre los conjuntos de permisos, consulte los conjuntos de permisos en la AWS IAM Identity Center Guía del usuario.
- Permisos de IAM usuario temporales: un IAM usuario o rol puede asumir un IAM rol para asumir temporalmente diferentes permisos para una tarea específica.
- Acceso multicuenta: puedes usar un IAM rol para permitir que alguien (un responsable de confianza) de una cuenta diferente acceda a los recursos de tu cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos Servicios de AWS, puede

adjuntar una política directamente a un recurso (en lugar de utilizar un rol como proxy). Para saber la diferencia entre las funciones y las políticas basadas en recursos para el acceso multicuenta, consulta el tema sobre el acceso a los recursos entre cuentas IAM en la Guía del IAM usuario.

- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un IAM usuario o un rol para realizar acciones en AWS, se le considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FASutiliza los permisos del director que llama a un Servicio de AWS, combinado con la solicitud Servicio de AWS para realizar solicitudes a los servicios intermedios. FASlas solicitudes solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o recursos para completar. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información detallada sobre la política a la hora de realizar FAS solicitudes, consulte Reenviar las sesiones de acceso.
  - Función de servicio: una función de servicio es una IAMfunción que un servicio asume para realizar acciones en su nombre. Un IAM administrador puede crear, modificar y eliminar un rol de servicio desde dentroIAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Función vinculada a un servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un Servicio de AWS. El servicio puede asumir la función de realizar una acción en su nombre. Los roles vinculados al servicio aparecen en su Cuenta de AWS y son propiedad del servicio. Un IAM administrador puede ver, pero no editar, los permisos de las funciones vinculadas al servicio.
- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un IAM rol para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y se están creando AWS CLI o AWS APIsolicitudes. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS Un rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia que se adjunte a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte Uso de un IAM rol para conceder permisos a aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del IAM usuario.

Para saber si se deben usar IAM roles o IAM usuarios, consulte Cuándo crear un IAM rol (en lugar de un usuario) en la Guía del IAM usuario.

Administración de acceso mediante políticas

Usted controla el acceso en AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto en AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan en AWS como JSON documentos. Para obtener más información sobre la estructura y el contenido de los documentos de JSON políticas, consulte Descripción general de JSON las políticas en la Guía del IAM usuario.

Los administradores pueden utilizar AWS JSONpolíticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder a los usuarios permiso para realizar acciones en los recursos que necesitan, un IAM administrador puede crear IAM políticas. A continuación, el administrador puede añadir las IAM políticas a las funciones y los usuarios pueden asumir las funciones.

IAMlas políticas definen los permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre su función en AWS Management Console, el AWS CLI, o el AWS API.

Políticas basadas en identidad

Las políticas basadas en la identidad son documentos de política de JSON permisos que se pueden adjuntar a una identidad, como un IAM usuario, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener información sobre cómo crear una política basada en la identidad, consulte Creación de IAM políticas en la Guía del usuario. IAM

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y funciones de su Cuenta de AWS. Las políticas gestionadas incluyen AWS las políticas gestionadas y las políticas gestionadas por el cliente. Para saber cómo elegir entre una política

gestionada o una política en línea, consulte Elegir entre políticas gestionadas y políticas integradas en la Guía del IAMusuario.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de JSON política que se adjuntan a un recurso. Algunos ejemplos de políticas basadas en recursos son las políticas de confianza de IAM roles y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar AWS políticas gestionadas desde una política basada IAM en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLsson similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de JSON políticas.

Amazon S3, AWS WAF, y Amazon VPC son ejemplos de servicios que admitenACLs. Para obtener más informaciónACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una función avanzada en la que se establecen los permisos máximos que una política basada en la identidad puede conceder a una IAM entidad (IAMusuario o rol). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites

de los permisos, consulte los límites de los permisos para IAM las entidades en la Guía del IAMusuario.

- Políticas de control de servicios (SCPs): SCPs son JSON políticas que especifican los permisos máximos para una organización o unidad organizativa (OU) en AWS Organizations. AWS Organizations es un servicio para agrupar y administrar de forma centralizada múltiples Cuentas de AWS que es propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar las políticas de control de servicios (SCPs) a cualquiera de tus cuentas o a todas ellas. SCPLimita los permisos de las entidades en las cuentas de los miembros, incluidas todas Usuario raíz de la cuenta de AWS. Para obtener más información acerca de OrganizationsSCPs, consulte Políticas de control de servicios en AWS Organizations Guía del usuario.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información, consulte las políticas de sesión en la Guía del IAM usuario.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información sobre cómo AWS determina si se permite una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del IAM usuario.

Cómo funciona EMR Serverless con IAM

Antes de administrar el acceso IAM a Amazon EMR Serverless, infórmese sobre IAM las funciones disponibles para su uso con Amazon EMR Serverless.

IAMfunciones que puede usar con Serverless EMR

IAMcaracterística	Soporte para Amazon EMR Serverless
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí

IAMcaracterística	Soporte para Amazon EMR Serverless
Recursos de políticas	Sí
Claves de condición de política	No
ACLs	No
ABAC(etiquetas en las políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	No
Roles vinculados al servicio	Sí

Para obtener una visión de alto nivel de cómo EMR Serverless y otros AWS los servicios funcionan con la mayoría de las IAM funciones, consulte AWS servicios con los que funcionan IAM en la Guía IAM del usuario.

Políticas basadas en la identidad para Serverless EMR

Compatibilidad con las políticas basadas en identidad: sí

Las políticas basadas en la identidad son documentos de política de JSON permisos que se pueden adjuntar a una identidad, como un IAM usuario, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener información sobre cómo crear una política basada en la identidad, consulte Creación de IAM políticas en la Guía del usuario. IAM

Con las políticas IAM basadas en la identidad, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está adjunto. Para obtener más información sobre todos los elementos que puede utilizar en una JSON política, consulte la referencia sobre los elementos de la IAM JSON política en la Guía del IAMusuario.

Ejemplos de políticas basadas en la identidad para Serverless EMR

Para ver ejemplos de políticas basadas en la identidad de Amazon EMR Serverless, consulte. Ejemplos de políticas basadas en la identidad para Serverless EMR

Políticas basadas en recursos dentro de Serverless EMR

Admite políticas basadas en recursos: no

Las políticas basadas en recursos son documentos de políticas que se JSON adjuntan a un recurso. Algunos ejemplos de políticas basadas en recursos son las políticas de confianza de IAM roles y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Para habilitar el acceso entre cuentas, puede especificar una cuenta completa o IAM entidades de otra cuenta como principales en una política basada en recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el IAM administrador de la cuenta de confianza también debe conceder permiso a la entidad principal (usuario o rol) para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para obtener más información, consulte el tema Acceso a recursos entre cuentas IAM en la Guía del IAM usuario.

Acciones políticas para EMR Serverless

Compatibilidad con las acciones de política: sí

Los administradores pueden usar AWS JSONpolíticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El `Action` elemento de una JSON política describe las acciones que puede utilizar para permitir o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que las asociadas AWS APIoperación. Hay algunas excepciones, como las acciones que solo requieren permisos y que no tienen una operación coincidente. API También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de acciones EMR sin servidor, consulte Acciones, recursos y claves de condición de Amazon EMR Serverless en la Referencia de autorización de servicios.

Las acciones políticas en EMR Serverless utilizan el siguiente prefijo antes de la acción.

```
emr-serverless
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Para ver ejemplos de políticas basadas en la identidad de Amazon EMR Serverless, consulte. Ejemplos de políticas basadas en la identidad para Serverless EMR

Recursos de políticas para Serverless EMR

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar AWS JSONpolíticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` JSON de política especifica el objeto o los objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso mediante su nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de recursos de Amazon EMR Serverless y sus tiposARNs, consulte Recursos definidos por Amazon EMR Serverless en la Referencia de autorización de servicios. Para saber qué acciones puede especificar para cada recurso, consulte Acciones, recursos y claves de condición de Amazon EMR Serverless. ARN

Para ver ejemplos de políticas basadas en la identidad de Amazon EMR Serverless, consulte. Ejemplos de políticas basadas en la identidad para Serverless EMR

Claves de condición de la política para Serverless EMR

Admite claves de condición de políticas específicas del servicio	No

Los administradores pueden usar AWS JSONpolíticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios `Condition` elementos en una declaración o varias claves en un solo `Condition` elemento, AWS los evalúa mediante una `AND` operación lógica. Si especifica varios valores para una sola clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder a un IAM usuario permiso para acceder a un recurso solo si está etiquetado con su nombre de IAM usuario. Para obtener más información, consulte los elementos de IAM política: variables y etiquetas en la Guía del IAM usuario.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas AWS claves de condición globales, consulte AWS claves de contexto de condiciones globales en la Guía IAM del usuario.

Para ver una lista de claves de condición de Amazon EMR Serverless y saber qué acciones y recursos puede usar una clave de condición, consulte Acciones, recursos y claves de condición para Amazon EMR Serverless en la Referencia de autorización de servicios.

Todas las EC2 acciones de Amazon admiten las claves de `ec2:Region` condición `aws:RequestedRegion` y. Para obtener más información, consulta Ejemplo: restringir el acceso a una región específica.

Listas de control de acceso (ACLs) en EMR Serverless

SoportaACLs: No

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLsson similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de JSON políticas.

Control de acceso basado en atributos () con Serverless ABAC EMR

Soporta ABAC (etiquetas en las políticas)	Sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define los permisos en función de los atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a IAM entidades (usuarios o roles) y a muchas AWS recursos. Etiquetar entidades y recursos es el primer paso deABAC. Luego, diseñe ABAC políticas para permitir las operaciones cuando la etiqueta del principal coincida con la etiqueta del recurso al que está intentando acceder.

ABACes útil en entornos de rápido crecimiento y ayuda en situaciones en las que la administración de políticas se vuelve engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información al respectoABAC, consulte ¿Qué es? ABAC en la Guía IAM del usuario. Para ver un tutorial con los pasos de configuraciónABAC, consulte Usar el control de acceso basado en atributos (ABAC) en la Guía del IAMusuario.

Uso de credenciales temporales con Serverless EMR

Compatibilidad con credenciales temporales: sí

Alguno Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluyendo qué Servicios de AWS trabaje con credenciales temporales, consulte Servicios de AWS que funcionan IAM en la Guía IAM del usuario.

Está utilizando credenciales temporales si inicia sesión en el AWS Management Console utilizando cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de su empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para obtener más información sobre el cambio de rol, consulte Cambiar a un rol (consola) en la Guía del IAMusuario.

Puede crear credenciales temporales manualmente mediante el AWS CLI o AWS API. A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM.

Permisos principales entre servicios para Serverless EMR

Admite sesiones de acceso directo (FAS): Sí

Cuando utiliza un IAM usuario o un rol para realizar acciones en AWS, se le considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FASutiliza los permisos del director que llama a un Servicio de AWS, combinado con la solicitud Servicio de AWS para realizar solicitudes a los servicios intermedios. FASlas solicitudes solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o recursos para completar. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información detallada sobre la política a la hora de realizar FAS solicitudes, consulte Reenviar las sesiones de acceso.

Funciones de servicio para EMR Serverless

Compatible con roles de servicio	No

Funciones vinculadas a servicios para Serverless EMR

Compatible con roles vinculados al servicio	Sí

Para obtener más información sobre la creación o la administración de funciones vinculadas a servicios, consulte AWS servicios con los que funcionan. IAM Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Uso de roles vinculados a servicios para Serverless EMR

Usos de Amazon EMR Serverless AWS Identity and Access Management (IAM) roles vinculados al servicio. Un rol vinculado a un servicio es un tipo único de IAM rol que está vinculado directamente a Serverless. EMR EMRServerless predefine los roles vinculados al servicio e incluyen todos los permisos que el servicio requiere para llamar a otros AWS servicios en su nombre.

Un rol vinculado a un servicio facilita la configuración de EMR Serverless porque no es necesario añadir manualmente los permisos necesarios. EMRServerless define los permisos de sus funciones vinculadas al servicio y, a menos que se defina lo contrario, solo EMR Serverless puede asumir sus funciones. Los permisos definidos incluyen la política de confianza y la política de permisos, y esa política de permisos no se puede adjuntar a ninguna otra entidad. IAM

Solo es posible eliminar un rol vinculado a un servicio después de eliminar sus recursos relacionados. Esto protege sus recursos EMR sin servidor porque no puede eliminar inadvertidamente el permiso de acceso a los recursos.

Para obtener información sobre otros servicios que admiten funciones vinculadas a servicios, consulte AWS Servicios que funcionan con IAM y busca los servicios que tienen la palabra «Sí» en la columna Funciones vinculadas al servicio. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado a servicios en cuestión.

Permisos de roles vinculados al servicio para Serverless EMR

EMRServerless usa el rol vinculado al servicio denominado para permitirle llamar AWSServiceRoleForAmazonEMRServerless AWS APIsen tu nombre.

El rol AWSServiceRoleForAmazonEMRServerless vinculado al servicio confía en los siguientes servicios para asumir el rol:

- `ops.emr-serverless.amazonaws.com`

La política de permisos de roles denominada `AmazonEMRServerlessServiceRolePolicy` permite a EMR Serverless realizar las siguientes acciones en los recursos especificados.

> **Note**
>
> El contenido de la política administrada cambia, por lo que la política que se muestra aquí puede estar desactualizada. Vea la mayoría up-to-date de las políticas A mazonEMRServerless ServiceRolePolicy en la AWS Management Console.

- Acción: `ec2:CreateNetworkInterface`
- Acción: `ec2:DeleteNetworkInterface`
- Acción: `ec2:DescribeNetworkInterfaces`
- Acción: `ec2:DescribeSecurityGroups`
- Acción: `ec2:DescribeSubnets`
- Acción: `ec2:DescribeVpcs`
- Acción: `ec2:DescribeDhcpOptions`
- Acción: `ec2:DescribeRouteTables`
- Acción: `cloudwatch:PutMetricData`

La siguiente es la `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"
                    ]
                }
            }
        }
    ]
}
```

La siguiente política de confianza se adjunta a esta función para permitir que el director de EMR Serverless asuma esta función.

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

Debe configurar los permisos para permitir que una IAM entidad (como un usuario, un grupo o un rol) cree, edite o elimine un rol vinculado a un servicio. Para obtener más información, consulte los permisos de los roles vinculados a un servicio en la Guía del usuario. IAM

Crear un rol vinculado a un servicio para Serverless EMR

No necesita crear manualmente un rol vinculado a servicios. Al crear una nueva aplicación EMR sin servidor en el AWS Management Console (con EMR Studio), el AWS CLI, o el AWS API, EMR Serverless crea el rol vinculado al servicio por usted. Debe configurar los permisos para permitir que una IAM entidad (como un usuario, un grupo o un rol) cree, edite o elimine un rol vinculado a un servicio.

Para crear el rol vinculado al AWSServiceRoleForAmazonEMRServerless servicio mediante IAM

Agregue la siguiente declaración a la política de permisos de la IAM entidad que necesita crear el rol vinculado al servicio.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Si elimina este rol vinculado a servicios y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Al crear una nueva aplicación EMR sin servidor, EMR Serverless vuelve a crear el rol vinculado al servicio automáticamente.

También puede usar la IAM consola para crear un rol vinculado a un servicio con el caso de uso de Serverless. EMR En el navegador AWS CLI o el AWS API, cree un rol vinculado al servicio con el nombre del `ops.emr-serverless.amazonaws.com` servicio. Para obtener más información, consulte Creación de un rol vinculado a un servicio en la Guía del usuario. IAM Si elimina este rol vinculado al servicio, puede utilizar este mismo proceso para volver a crear el rol.

Edición de un rol vinculado a un servicio para Serverless EMR

EMRServerless no permite editar el rol AWSServiceRoleForAmazonEMRServerless vinculado al servicio porque varias entidades pueden hacer referencia al rol. No puede editar el AWS: IAM política propia que utiliza el rol vinculado al servicio EMR Serverless, ya que contiene todos los permisos necesarios que Serverless necesita. EMR Sin embargo, puede editar la descripción del rol utilizando. IAM

Para editar la descripción del rol AWSServiceRoleForAmazonEMRServerless vinculado al servicio mediante IAM

Agregue la siguiente declaración a la política de permisos de la IAM entidad que necesita editar la descripción de un rol vinculado al servicio.

```
{
    "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 obtener más información, consulte Edición de un rol vinculado a un servicio en la Guía del usuario. IAM

Eliminar un rol vinculado a un servicio para Serverless EMR

Si ya no necesita usar una característica o servicio que requieran un rol vinculado a un servicio, le recomendamos que elimine dicho rol. Esto es para que no tenga una entidad no utilizada que no esté monitoreada o mantenida activamente. Sin embargo, debe eliminar todas las aplicaciones EMR sin servidor de todas las regiones para poder eliminar la función vinculada al servicio.

> **Note**
>
> Si el servicio EMR Serverless utiliza el rol al intentar eliminar los recursos asociados al rol, es posible que la eliminación no se realice correctamente. En tal caso, espere unos minutos e intente de nuevo la operación.

Para eliminar el rol vinculado al AWSServiceRoleForAmazonEMRServerless servicio mediante IAM

Agregue la siguiente declaración a la política de permisos de la IAM entidad que necesita eliminar un rol vinculado al servicio.

```
{
    "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 eliminar manualmente el rol vinculado al servicio mediante IAM

Utilice la IAM consola, la AWS CLI, o el AWS APIpara eliminar el rol AWSServiceRoleForAmazonEMRServerless vinculado al servicio. Para obtener más información, consulte Eliminar un rol vinculado a un servicio en la Guía del usuario. IAM

Regiones compatibles con funciones vinculadas a servicios EMR sin servidor

EMRServerless admite el uso de funciones vinculadas al servicio en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte AWS Regiones y puntos finales.

Funciones de tiempo de ejecución de trabajos para Amazon EMR Serverless

Puede especificar los permisos de IAM rol que puede asumir un trabajo EMR sin servidor al llamar a otros servicios en su nombre. Esto incluye el acceso a Amazon S3 para cualquier fuente de datos, destino y otros AWS recursos como clústeres de Amazon Redshift y tablas de DynamoDB. Para obtener más información sobre cómo crear un rol, consulte. Cree un rol de ejecución de tareas

Ejemplos de políticas de tiempo de ejecución

Puede adjuntar una política de tiempo de ejecución, como la siguiente, a un rol de tiempo de ejecución de un trabajo. La siguiente política de tiempo de ejecución de tareas permite:

- Lea el acceso a los buckets de Amazon S3 con EMR ejemplos.
- Acceso completo a los buckets de S3.
- Cree y lea el acceso a AWS Catálogo de datos de Glue.

Para añadir acceso a otros AWS recursos como DynamoDB, tendrás que incluir sus permisos en la política al crear el rol de tiempo de ejecución.

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

Transfiera los privilegios del rol

Puede adjuntar políticas de IAM permisos al rol de un usuario para que el usuario pueda transferir solo los roles aprobados. Esto permite a los administradores controlar qué usuarios pueden transferir funciones específicas de ejecución de tareas a tareas EMR sin servidor. Para obtener más información sobre la configuración de permisos, consulte Otorgar permisos a un usuario para transferir un rol a un AWS servicio.

El siguiente es un ejemplo de política que permite transferir una función de ejecución de tareas al director del servicio EMR Serverless.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

Ejemplos de políticas de acceso de usuarios para Serverless EMR

Puede configurar políticas detalladas para sus usuarios en función de las acciones que desee que realice cada usuario al interactuar con las aplicaciones sin servidor. EMR Las siguientes políticas son ejemplos que pueden ayudar a configurar los permisos correctos para sus usuarios. Esta sección se centra únicamente en las políticas EMR sin servidor. Para ver ejemplos de las políticas de usuario de EMR Studio, consulte Configurar los permisos de usuario de EMR Studio. Para obtener información sobre cómo adjuntar políticas a IAM los usuarios (principales), consulte Administrar IAM políticas en la Guía del IAM usuario.

Política para usuarios avanzados

Para conceder todas las acciones necesarias a EMR Serverless, cree y adjunte una `AmazonEMRServerlessFullAccess` política al IAM usuario, rol o grupo requerido.

El siguiente es un ejemplo de política que permite a los usuarios avanzados crear y modificar aplicaciones EMR sin servidor, así como realizar otras acciones, como enviar y depurar tareas. Revela todas las acciones que EMR Serverless requiere para otros servicios.

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

```
}
```

Cuando habilita la conectividad de red con sus VPC aplicaciones EMR sin servidor crean interfaces de red EC2 elásticas de Amazon (ENIs) para comunicarse con VPC los recursos. La siguiente política garantiza que las nuevas solo EC2 ENIs se creen en el contexto de las aplicaciones EMR sin servidor.

> **Note**
>
> Recomendamos encarecidamente establecer esta política para garantizar que los usuarios no puedan crear EC2 ENIs excepto en el contexto del lanzamiento de aplicaciones EMR sin 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"
                }
            }
        }
}
```

Si desea restringir el acceso EMR sin servidor a determinadas subredes, puede etiquetar cada subred con una condición de etiqueta. Esta IAM política garantiza que las aplicaciones EMR sin servidor solo se puedan crear EC2 ENIs dentro de las subredes 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**
>
> Si es un administrador o un usuario avanzado que está creando su primera aplicación, debe configurar sus políticas de permisos para poder crear un rol vinculado a un servicio EMR sin servidor. Para obtener más información, consulte Uso de roles vinculados a servicios para Serverless EMR.

La siguiente IAM política le permite crear un rol vinculado a un servicio EMR sin servidor para su cuenta.

```
{
   "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 ingeniería de datos

El siguiente es un ejemplo de política que permite a los usuarios permisos de solo lectura en aplicaciones EMR sin servidor, así como la posibilidad de enviar y depurar trabajos. Tenga en cuenta que, dado que esta política no deniega acciones explícitamente, se puede seguir utilizando una instrucción de política distinta para otorgar acceso a acciones especificadas.

```
{
```

```
    "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 etiquetas para el control de acceso

Puede utilizar las condiciones de etiqueta para un control de acceso detallado. Por ejemplo, puedes restringir a los usuarios de un equipo para que solo puedan enviar trabajos a aplicaciones EMR sin servidor etiquetadas con el nombre de su equipo.

```
{
    "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 el control de acceso basado en etiquetas

Puede utilizar las condiciones de su política basada en la identidad para controlar el acceso a las aplicaciones y las ejecuciones de tareas en función de las etiquetas.

Los siguientes ejemplos muestran diferentes escenarios y formas de utilizar los operadores de condición con claves de condición EMR sin servidor. Estas declaraciones IAM de política están destinadas únicamente a fines de demostración y no deben utilizarse en entornos de producción. Existen varias maneras de combinar las instrucciones de políticas para conceder y denegar permisos de acuerdo con sus requisitos. Para obtener más información sobre IAM las políticas de planificación y prueba, consulte la Guía del IAM usuario.

> **Important**
>
> La denegación de permisos explícita para acciones de etiquetado de acciones es un factor importante. Esto impide que los usuarios etiqueten un recurso y, de esta forma, se concedan a sí mismos permisos que usted no tenía previsto conceder. Si no se deniegan las acciones de etiquetado de un recurso, el usuario puede modificar las etiquetas y eludir la intención de las políticas basadas en etiquetas. Para ver un ejemplo de una política que deniega las acciones de etiquetado, consulte Denegar el acceso para agregar y eliminar etiquetas.

Los ejemplos siguientes muestran las políticas de permisos basadas en la identidad que se utilizan para controlar las acciones que se permiten en las aplicaciones sin EMR servidor.

Permitir acciones solo en recursos con valores de etiqueta específicos

En el siguiente ejemplo de política, el operador de `StringEquals` condición intenta coincidir `dev` con el valor del departamento de etiquetas. Si el departamento de etiquetas no se ha agregado a la aplicación o no contiene el valor`dev`, la política no se aplica y esta política no permite las acciones. Si ninguna otra declaración de política permite estas acciones, el usuario solo podrá trabajar con aplicaciones que tengan esta etiqueta con este valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

También puede especificar varios valores de etiqueta utilizando un operador de condición. Por ejemplo, para permitir acciones en aplicaciones en las que la `department` etiqueta contiene el valor `devtest`, puede reemplazar el bloque de condiciones del ejemplo anterior por el siguiente.

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

Requerir etiquetado cuando se crea un recurso

En el ejemplo siguiente, la etiqueta debe aplicarse al crear la aplicación.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

La siguiente declaración de política permite a un usuario crear una aplicación solo si la aplicación tiene una `department` etiqueta, que puede contener cualquier valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "emr-serverless:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

Denegar el acceso para agregar y eliminar etiquetas

Esta política impide que un usuario agregue o elimine etiquetas en las aplicaciones EMR sin servidor con una `department` etiqueta cuyo valor no `dev` sea el indicado.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
```

```
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Ejemplos de políticas basadas en la identidad para Serverless EMR

De forma predeterminada, los usuarios y los roles no tienen permiso para crear o modificar los recursos de Amazon EMR Serverless. Tampoco pueden realizar tareas mediante el AWS Management Console, AWS Command Line Interface (AWS CLI), o AWS API. Para conceder a los usuarios permiso para realizar acciones en los recursos que necesitan, un IAM administrador puede crear IAM políticas. A continuación, el administrador puede añadir las IAM políticas a las funciones y los usuarios pueden asumir las funciones.

Para obtener información sobre cómo crear una política IAM basada en la identidad mediante estos documentos de JSON política de ejemplo, consulte Creación de IAM políticas en la Guía del IAMusuario.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Amazon EMR Serverless, incluido el ARNs formato de cada uno de los tipos de recursos, consulte Acciones, recursos y claves de condición de Amazon EMR Serverless en la Referencia de autorización de servicios.

Temas

- Prácticas recomendadas sobre las políticas
- Cómo permitir a los usuarios consultar sus propios permisos

Prácticas recomendadas sobre las políticas

> **Note**
>
> EMRServerless no admite políticas administradas, por lo que la primera práctica que se detalla a continuación no es aplicable.

Las políticas basadas en la identidad determinan si alguien puede crear recursos de Amazon EMR Serverless de su cuenta, acceder a ellos o eliminarlos. Estas acciones pueden suponer costes para su Cuenta de AWS. Al crear o editar políticas basadas en la identidad, siga estas directrices y recomendaciones:

- Comience con AWS políticas gestionadas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice la AWS políticas gestionadas que conceden permisos para muchos casos de uso habituales. Están disponibles en su Cuenta de AWS. Le recomendamos que reduzca aún más los permisos definiendo AWS políticas gestionadas por el cliente que sean específicas para sus casos de uso. Para obtener más información, consulte AWS políticas gestionadas o AWS políticas gestionadas para las funciones laborales en la Guía IAM del usuario.
- Aplique permisos con privilegios mínimos: cuando establezca permisos con IAM políticas, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Para obtener más información sobre cómo IAM aplicar permisos, consulte Políticas y permisos IAM en la IAM Guía del usuario.
- Utilice las condiciones en IAM las políticas para restringir aún más el acceso: puede añadir una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de política para especificar que todas las solicitudes deben enviarse medianteSSL. También puede utilizar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de un procedimiento específico Servicio de AWS, como, por ejemplo, AWS CloudFormation. Para obtener más información, consulte los elementos IAM JSON de la política: Condición en la Guía del IAM usuario.
- Utilice IAM Access Analyzer para validar sus IAM políticas y garantizar permisos seguros y funcionales: IAM Access Analyzer valida las políticas nuevas y existentes para que se ajusten al lenguaje de las políticas (JSON) y IAM a las IAM mejores prácticas. IAMAccess Analyzer proporciona más de 100 comprobaciones de políticas y recomendaciones prácticas para ayudarle

a crear políticas seguras y funcionales. Para obtener más información, consulte la validación de políticas de IAM Access Analyzer en la Guía del IAM usuario.

- Requerir autenticación multifactorial (MFA): si tiene un escenario que requiere IAM usuarios o un usuario raíz en su Cuenta de AWS, actívala MFA para mayor seguridad. Para solicitarlo MFA cuando se cancelen API las operaciones, añada MFA condiciones a sus políticas. Para obtener más información, consulte Configuración del API acceso MFA protegido en la Guía del IAM usuario.

Para obtener más información sobre las prácticas recomendadasIAM, consulte las prácticas recomendadas de seguridad IAM en la Guía del IAM usuario.

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo se muestra cómo se puede crear una política que permita a IAM los usuarios ver las políticas integradas y administradas asociadas a su identidad de usuario. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante el AWS CLI o 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": "*"
        }
    ]
}
```

Amazon EMR Serverless actualiza a AWS políticas administradas

Ver detalles sobre las actualizaciones de AWS administró políticas para Amazon EMR Serverless desde que este servicio comenzó a rastrear estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbase al RSS feed de la página del historial de documentos de Amazon EMR Serverless.

Cambio	Descripción	Fecha
R mazonEMRServerless ServiceRolePolicy — Actualización de una política existente	Amazon EMR Serverles s agregó la nueva `Sid` `CloudWatchPolicySt atement` y `EC2Policy Statement` a la mazonEMRServerless ServiceRolePolicy política A.	25 de enero de 2024
R mazonEMRServerless ServiceRolePolicy : Actualización de una política existente	Amazon EMR Serverless agregó nuevos permisos para permitir a Amazon EMR Serverless publicar métricas de cuenta agregadas para el CPU uso de v en el `"AWS/Usage"` espacio de nombres.	20 de abril de 2023

Cambio	Descripción	Fecha
Amazon EMR Serverless comenzó a rastrear los cambios	Amazon EMR Serverless comenzó a rastrear los cambios en su AWS políticas gestionadas.	20 de abril de 2023

Solución de problemas de identidad y acceso a Amazon EMR Serverless

Utilice la siguiente información como ayuda para diagnosticar y solucionar problemas comunes que pueden surgir al trabajar con Amazon EMR Serverless yIAM.

Temas

- No estoy autorizado a realizar ninguna acción en Amazon EMR Serverless
- No estoy autorizado a realizar lo siguiente: PassRole
- Quiero permitir que personas ajenas a mi AWS cuenta para acceder a mis recursos de Amazon EMR Serverless

No estoy autorizado a realizar ninguna acción en Amazon EMR Serverless

Si el archivo de AWS Management Console le indica que no está autorizado a realizar una acción, por lo que debe ponerse en contacto con su administrador para obtener ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

En el siguiente ejemplo, el error se produce cuando el usuario `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `emr-serverless:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless:GetWidget on resource: my-example-widget
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *`my-example-widget`* mediante la acción `emr-serverless:`*`GetWidget`*.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no está autorizado a realizar la `iam:PassRole` acción, sus políticas deben actualizarse para que pueda transferir un rol a Amazon EMR Serverless.

Alguno Servicios de AWS le permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

El siguiente ejemplo de error se produce cuando un IAM usuario llamado `marymajor` intenta usar la consola para realizar una acción en Amazon EMR Serverless. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mi AWS cuenta para acceder a mis recursos de Amazon EMR Serverless

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admiten políticas basadas en recursos o listas de control de acceso (ACLs), puede usar esas políticas para permitir que las personas accedan a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si Amazon EMR Serverless admite estas funciones, consulteIdentity and Access Management (IAM) en Amazon EMR Serverless.
- Para obtener información sobre cómo proporcionar acceso a sus recursos en Cuentas de AWS que te pertenezca, consulta Proporcionar acceso a un IAM usuario en otro Cuenta de AWS que le pertenezca en la Guía IAM del usuario.

- Para obtener información sobre cómo proporcionar acceso a sus recursos a terceros Cuentas de AWS, consulte Proporcionar acceso a Cuentas de AWS propiedad de terceros en la Guía IAM del usuario.
- Para obtener información sobre cómo proporcionar acceso mediante la federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (federación de identidades) en la Guía del IAM usuario.
- Para saber la diferencia entre el uso de roles y políticas basadas en recursos para el acceso entre cuentas, consulte el acceso a recursos entre cuentas IAM en la Guía del usuario. IAM

Uso de EMR Serverless con AWS Lake Formation para un control de acceso detallado

Información general

Con las EMR versiones 7.2.0 y posteriores de Amazon, puede aprovechar AWS Lake Formation para aplicar controles de acceso detallados a las tablas del catálogo de datos respaldadas por S3. Esta capacidad le permite configurar los controles de acceso a nivel de tabla, fila, columna y celda para read consultas dentro de sus trabajos de Amazon EMR Serverless Spark. Para configurar un control de acceso detallado para los trabajos por lotes y las sesiones interactivas de Apache Spark, utilice Studio. EMR Consulte las siguientes secciones para obtener más información sobre Lake Formation y cómo usarlo con EMR Serverless.

Uso de Amazon EMR Serverless con AWS Lake Formation incurre en cargos adicionales. Para obtener más información, consulta los EMRprecios de Amazon.

Cómo funciona EMR Serverless con AWS Lake Formation

El uso de EMR Serverless con Lake Formation te permite aplicar una capa de permisos en cada trabajo de Spark para aplicar el control de permisos de Lake Formation cuando EMR Serverless ejecuta trabajos. EMRServerless usa los perfiles de recursos de Spark para crear dos perfiles para ejecutar los trabajos de forma eficaz. El perfil de usuario ejecuta el código proporcionado por el usuario, mientras que el perfil del sistema aplica las políticas de Lake Formation. Para obtener más información, consulte ¿Qué es AWS Lake Formationy Consideraciones y limitaciones.

Cuando utilices la capacidad preinicializada con Lake Formation, te recomendamos que tengas un mínimo de dos controladores Spark. Cada trabajo habilitado para Lake Formation utiliza dos

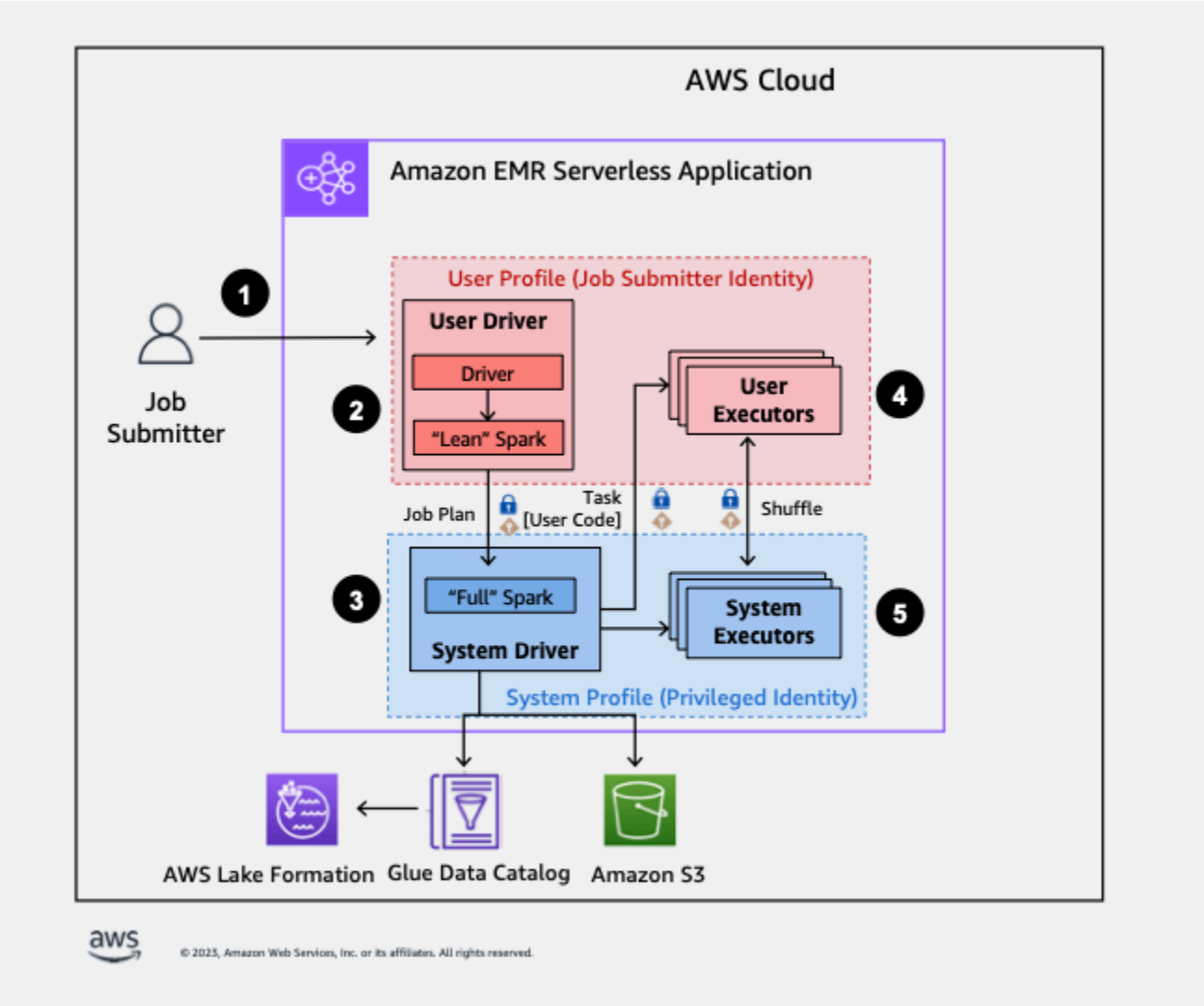
controladores Spark, uno para el perfil de usuario y otro para el perfil del sistema. Para obtener el mejor rendimiento, debería utilizar el doble de controladores para los trabajos habilitados para Lake Formation en comparación con si no utilizara Lake Formation.

Cuando ejecutes trabajos de Spark en EMR Serverless, también debes tener en cuenta el impacto de la asignación dinámica en la administración de recursos y el rendimiento de los clústeres. La configuración `spark.dynamicAllocation.maxExecutors` del número máximo de ejecutores por perfil de recurso se aplica tanto a los ejecutores de usuario como a los ejecutores del sistema. Si configura ese número para que sea igual al número máximo permitido de ejecutores, es posible que la ejecución de la tarea se bloquee debido a que un tipo de ejecutor utiliza todos los recursos disponibles, lo que impide que el otro ejecutor ejecute las tareas.

Para no quedarse sin recursos, EMR Serverless establece el número máximo predeterminado de ejecutores por perfil de recurso en el 90% del valor. `spark.dynamicAllocation.maxExecutors` Puede anular esta configuración si la especifica `spark.dynamicAllocation.maxExecutorsRatio` con un valor entre 0 y 1. Además, también puede configurar las siguientes propiedades para optimizar la asignación de recursos y el rendimiento general:

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

La siguiente es una descripción general de alto nivel de cómo EMR Serverless obtiene acceso a los datos protegidos por las políticas de seguridad de Lake Formation.



1. Un usuario envía un trabajo de Spark a un AWS Lake Formation-Aplicación EMR sin servidor habilitada.
2. EMRServerless envía el trabajo a un controlador de usuario y lo ejecuta en el perfil de usuario. El controlador de usuario ejecuta una versión sencilla de Spark que no permite lanzar tareas, solicitar ejecutores ni acceder a S3 ni al catálogo de Glue. Crea un plan de trabajo.
3. EMRServerless configura un segundo controlador denominado controlador del sistema y lo ejecuta en el perfil del sistema (con una identidad privilegiada). EMRServerless configura un TLS canal cifrado entre los dos controladores para la comunicación. El controlador de usuario utiliza el canal para enviar los planes de trabajo al controlador del sistema. El controlador del sistema no ejecuta el código enviado por el usuario. Ejecuta Spark a pleno rendimiento y se comunica con S3 y con el catálogo de datos para acceder a los datos. Solicita ejecutores y compila el Job Plan en una secuencia de etapas de ejecución.

4. EMRLuego, Serverless ejecuta las etapas en los ejecutores con el controlador de usuario o el controlador del sistema. En cualquier etapa, el código de usuario se ejecuta exclusivamente en los ejecutores de perfiles de usuario.
5. Etapas que leen datos de las tablas del catálogo de datos protegidas por AWS Lake Formation o las que aplican filtros de seguridad se delegan en los ejecutores del sistema.

Permitiendo la formación de lagos en Amazon EMR

Para habilitar Lake Formation, debe configurarlo en una `spark-defaults` clasificación `spark.emr-serverless.lakeformation.enabled` `true` inferior para el parámetro de configuración de tiempo de ejecución al crear una aplicación EMR sin 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"
```

También puede activar Lake Formation al crear una nueva aplicación en EMR Studio. Elija Use Lake Formation para un control de acceso detallado, disponible en Configuraciones adicionales.

El cifrado entre trabajadores está habilitado de forma predeterminada cuando usa Lake Formation con EMR Serverless, por lo que no tiene que volver a habilitar explícitamente el cifrado entre trabajadores.

Habilitando Lake Formation para Spark jobs

Para habilitar Lake Formation para trabajos individuales de Spark, establézcalo en `spark.emr-serverless.lakeformation.enabled` true cuando lo utilices`spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

IAMPermisos de rol en tiempo de ejecución de Job

Los permisos de Lake Formation controlan el acceso a AWS Los recursos del catálogo de datos de Glue, las ubicaciones de Amazon S3 y los datos subyacentes en esas ubicaciones. IAMlos permisos controlan el acceso a Lake Formation y AWS Glue APIs y recursos. Aunque es posible que tenga el permiso de Lake Formation para acceder a una tabla del catálogo de datos (SELECT), la operación fallará si no tiene el IAM permiso para la `glue:Get*` API operación.

El siguiente es un ejemplo de política sobre cómo proporcionar IAM permisos para acceder a un script en S3 mediante la carga de registros en S3, AWS APIPermisos de Glue y permiso para acceder a 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": ["*"]
        }
    ]
}
```

Configuración de los permisos de Lake Formation para el rol de ejecución de tareas

Primero, registre la ubicación de su mesa de colmenas con Lake Formation. A continuación, cree los permisos para el rol de ejecución de su trabajo en la tabla que desee. Para obtener más información sobre Lake Formation, consulte Qué es AWS Lake Formation? en el AWS Lake Formation Guía para desarrolladores.

Después de configurar los permisos de Lake Formation, puedes enviar trabajos de Spark en Amazon EMR Serverless. Para obtener más información sobre los trabajos de Spark, consulta los ejemplos de Spark.

Enviar una ejecución de tareas

Cuando termines de configurar las subvenciones de Lake Formation, podrás enviar trabajos a Spark en EMR Serverless. Para ejecutar los trabajos de Iceberg, debes proporcionar las siguientes `spark-submit` propiedades.

```
--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
```

Compatibilidad con el formato de tabla abierta

La EMR versión 7.2.0 de Amazon incluye soporte para un control de acceso detallado basado en Lake Formation. EMRServerless admite los tipos de tablas Hive e Iceberg. En la siguiente tabla se describen todas las operaciones admitidas.

Operaciones	Hive	Iceberg
DDLcomandos	Solo con permisos de IAM rol	Solo con permisos de IAM rol
Consultas incrementales	No aplicable	Totalmente compatible
Consultas de viaje en el tiempo	No se aplica a este formato de tabla	Totalmente compatible
Tabla de metadatos	No se aplica a este formato de tabla	Se admite, pero algunas tablas están ocultas. Consulte las consideraciones y limitaciones para obtener más información.
`DML INSERT`	Solo con IAM permisos	Solo con IAM permisos
DML UPDATE	No se aplica a este formato de tabla	Solo con IAM permisos
`DML DELETE`	No se aplica a este formato de tabla	Solo con IAM permisos
Operaciones de lectura	Totalmente compatible	Totalmente compatible
Procedimientos almacenados	No aplicable	Compatible con las excepciones de `register_table` y `migrate`. Consulte las consideraciones y limitaciones para obtener más información.

Consideraciones y limitaciones

Tenga en cuenta las siguientes consideraciones y limitaciones cuando utilice Lake Formation con EMR Serverless.

> **Note**
>
> Cuando habilitas Lake Formation para un trabajo de Spark en EMR Serverless, el trabajo lanza un controlador de sistema y un controlador de usuario. Si especificaste la capacidad preinicializada en el momento del lanzamiento, los controladores se aprovisionarán a partir de la capacidad preinicializada y el número de controladores del sistema será igual al número de controladores de usuario que especifiques. Si elige la capacidad bajo demanda, EMR Serverless lanza un controlador de sistema además de un controlador de usuario. Para estimar los costos asociados a su trabajo EMR sin servidor con Lake Formation, utilice la AWS Pricing Calculator.

Amazon EMR Serverless with Lake Formation está disponible en todas las regiones EMRsin servidor compatibles, excepto AWS GovCloud (Este de EE. UU.) y AWS GovCloud (EEUU-Oeste).

- Amazon EMR Serverless admite un control de acceso detallado a través de Lake Formation solo para las tablas Apache Hive y Apache Iceberg. Los formatos de Apache Hive incluyen Parquet y xSv. ORC
- Las aplicaciones habilitadas para Lake Formation no admiten el uso de imágenes personalizadas EMR sin servidor.
- No puedes dejar de trabajar `DynamicResourceAllocation` en Lake Formation.
- Solo puedes usar Lake Formation con trabajos de Spark.
- EMRServerless with Lake Formation solo admite una sesión de Spark durante un trabajo.
- EMRServerless with Lake Formation solo admite consultas de tablas entre cuentas compartidas a través de enlaces de recursos.
- No se admite lo siguiente:
  - Conjuntos de datos distribuidos resilientes () RDD
  - Transmisión de Spark
  - Permisos concedidos a Write with Lake Formation
  - Control de acceso para columnas anidadas

- EMRServerless bloquea las funcionalidades que podrían socavar el aislamiento total del controlador del sistema, incluidas las siguientes:
  - UDTsiveUDFs, H y cualquier función definida por el usuario que incluya clases personalizadas
  - Orígenes de datos personalizados
  - Suministro de tarros adicionales para la extensión, el conector o el metastore de Spark
  - `ANALYZE TABLE` command
- Para hacer cumplir los controles de acceso `EXPLAIN PLAN` y DDL las operaciones, como `DESCRIBE TABLE` no exponer información restringida.
- EMRServerless restringe el acceso a los registros de Spark del controlador del sistema en las aplicaciones habilitadas para Lake Formation. Dado que el controlador del sistema se ejecuta con más acceso, los eventos y registros que genera el controlador del sistema pueden incluir información confidencial. Para evitar que usuarios o códigos no autorizados accedan a estos datos confidenciales, EMR Serverless deshabilitó el acceso a los registros de los controladores del sistema. Para solucionar problemas, póngase en contacto con AWS soporte.
- Si ha registrado una ubicación de tabla en Lake Formation, la ruta de acceso a los datos pasa por las credenciales almacenadas de Lake Formation, independientemente del IAM permiso para el rol de ejecución de tareas EMR sin servidor. Si configura mal el rol registrado con la ubicación de la tabla, se producirá un error en los trabajos que se envíen en los que se utilice el rol con IAM permiso de S3 para la ubicación de la tabla.
- Escribir en una tabla de Lake Formation utiliza IAM permisos en lugar de los permisos concedidos por Lake Formation. Si el rol de ejecución de su trabajo tiene los permisos de S3 necesarios, puede usarlo para ejecutar operaciones de escritura.

Las siguientes son consideraciones y limitaciones a la hora de utilizar Apache Iceberg:

- Solo puede usar Apache Iceberg con el catálogo de sesiones y no con catálogos con nombres arbitrarios.
- Las tablas de iceberg que están registradas en Lake Formation solo admiten las tablas de metadatos `history`metadata_log_entries,`snapshots`,, `filesmanifests`, `yrefs`. Amazon EMR oculta las columnas que pueden contener datos confidenciales, como `partitionspath`, `ysummaries`. Esta limitación no se aplica a las tablas de iceberg que no estén registradas en Lake Formation.

- Las tablas que no se registran en Lake Formation admiten todos los procedimientos almacenados de Iceberg. Los `migrate` procedimientos `register_table` y no son compatibles con ninguna tabla.
- Le recomendamos que utilice Iceberg DataFrameWriter V2 en lugar de V1.

Resolución de problemas

Consulte las siguientes secciones para obtener soluciones de solución de problemas.

Registro

EMRServerless utiliza los perfiles de recursos de Spark para dividir la ejecución de los trabajos. EMRServerless usa el perfil de usuario para ejecutar el código que proporcionó, mientras que el perfil del sistema aplica las políticas de Lake Formation. Puede acceder a los registros de las tareas ejecutadas como perfil de usuario.

Interfaz de usuario en vivo y servidor de historial de Spark

La interfaz de usuario en vivo y el servidor de historial de Spark contienen todos los eventos de Spark generados a partir del perfil de usuario y los eventos redactados generados a partir del controlador del sistema.

Puedes ver todas las tareas del usuario y de los controladores del sistema en la pestaña Ejecutores. Sin embargo, los enlaces de registro solo están disponibles para el perfil de usuario. Además, parte de la información está redactada en la interfaz de usuario de Live, como el número de registros de salida.

Job falló con permisos de Lake Formation insuficientes

Asegúrese de que su rol de ejecución de tareas tenga los permisos para ejecutarse SELECT y estén DESCRIBE en la tabla a la que está accediendo.

Job con RDD ejecución fallida

EMRActualmente, Serverless no admite operaciones de conjuntos de datos distribuidos resilientes (RDD) en trabajos habilitados para Lake Formation.

No se puede acceder a los archivos de datos en Amazon S3

Asegúrese de haber registrado la ubicación del lago de datos en Lake Formation.

Excepción de validación de seguridad

EMRServerless detectó un error de validación de seguridad. Contacto AWS soporte de asistencia.

Uso compartido AWS Glue: catálogo de datos y tablas entre cuentas

Puede compartir bases de datos y tablas entre cuentas y seguir utilizando Lake Formation. Para obtener más información, consulte Intercambio de datos entre cuentas en Lake Formation y How do I share AWS Glue Data Catalog y tablas entre cuentas utilizando AWS Lake Formation?.

Cifrado entre trabajadores

Con EMR las versiones 6.15.0 y posteriores de Amazon, puedes habilitar la comunicación TLS cifrada mutua entre los trabajadores en tus tareas de Spark. Cuando está activado, EMR Serverless genera y distribuye automáticamente un certificado único para cada trabajador aprovisionado en el marco de tus ejecuciones de trabajo. Cuando estos trabajadores se comunican para intercambiar mensajes de control o transferir datos aleatorios, establecen una TLS conexión mutua y utilizan los certificados configurados para verificar la identidad de los demás. Si un trabajador no puede verificar otro certificado, se produce un error en el TLS protocolo de enlace y EMR Serverless interrumpe la conexión entre ellos.

Si utilizas Lake Formation con EMR Serverless, el TLS cifrado mutuo está activado de forma predeterminada.

Habilitar el TLS cifrado mutuo en Serverless EMR

Para habilitar el TLS cifrado mutuo en tu aplicación Spark, establécelo en true `spark.ssl.internode.enabled` al crear una aplicación EMR sin servidor. Si estás usando el AWS consola para crear una aplicación EMR sin servidor, selecciona Usar configuración personalizada, luego expande Configuración de la aplicación e ingresa tu`runtimeConfiguration`.

```
aws emr-serverless create-application \
--release-label emr-6.15.0 \
--runtime-configuration '{
  "classification": "spark-defaults",
  "properties": {"spark.ssl.internode.enabled": "true"}
}' \
--type "SPARK"
```

Si quieres habilitar el TLS cifrado mutuo para cada ejecución de tareas de Spark, establézcalo en `spark.ssl.internode.enabled` true cuando lo utilices`spark-submit`.

```
--conf spark.ssl.internode.enabled=true
```

Secrets Manager para la protección de datos con EMR Serverless

AWS Secrets Manager es un servicio de almacenamiento secreto que puede utilizar para proteger las credenciales, las API claves y otra información secreta de la base de datos. Luego, en tu código, puedes reemplazar las credenciales codificadas por una API llamada a Secrets Manager. Esto ayuda a garantizar que el secreto no se vea comprometido por alguien que examine tu código, ya que el secreto no está ahí. Para obtener una descripción general, consulta la AWS Secrets Manager Guía del usuario.

Secrets Manager cifra los secretos mediante AWS Key Management Service claves. Para obtener más información, consulte Cifrado y descifrado secretos en AWS Secrets Manager Guía del usuario.

Puede configurar Secrets Manager para rotar el secreto automáticamente de acuerdo con la programación que especifique. Esto le permite reemplazar secretos a largo plazo con secretos a corto plazo, lo que contribuye a reducir significativamente el riesgo de peligro. Para obtener más información, consulte Rotar AWS Secrets Manager secretos en el AWS Secrets Manager Guía del usuario.

Amazon EMR Serverless se integra con AWS Secrets Manager para que pueda almacenar sus datos en Secrets Manager y utilizar el ID secreto en sus configuraciones.

Cómo usa EMR Serverless los secretos

Cuando guardas tus datos en Secrets Manager y utilizas el ID secreto en tus configuraciones de EMR Serverless, no pasas datos de configuración confidenciales a EMR Serverless en texto plano ni los expones a fuentes externas. APIs Si indica que un par clave-valor contiene un ID secreto para un secreto que ha almacenado en Secrets Manager, EMR Serverless recupera el secreto cuando envía los datos de configuración a los trabajadores para que ejecuten las tareas.

Para indicar que un par clave-valor de una configuración contiene una referencia a un secreto almacenado en Secrets Manager, añada la `EMR.secret@` anotación al valor de la configuración. Para cualquier propiedad de configuración con anotación de ID secreta, EMR Serverless llama a Secrets Manager y resuelve el secreto en el momento de la ejecución del trabajo.

¿Cómo crear un secreto

Para crear un secreto, siga los pasos que se indican en Crear un AWS Secrets Manager secreto en el AWS Secrets Manager Guía del usuario. En el paso 3, selecciona el campo de texto sin formato para introducir tu valor confidencial.

Proporcione un secreto en una clasificación de configuración

Los siguientes ejemplos muestran cómo proporcionar un secreto en una clasificación de configuración en`StartJobRun`. Si desea configurar las clasificaciones de Secrets Manager a nivel de aplicación, consulteConfiguración de aplicaciones predeterminada para Serverless EMR.

En los ejemplos, *`SecretName`* sustitúyalo por el nombre del secreto que se va a recuperar. Incluye el guion seguido de los seis caracteres que Secrets Manager añade al final del secretoARN. Para obtener más información, consulte ¿Cómo crear un secreto.

En esta sección

- Especifica las referencias secretas: Spark
- Especifique las referencias secretas: Hive

Especifica las referencias secretas: Spark

Example — Especifica las referencias secretas en la configuración externa del metaalmacén de Hive para 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 las referencias secretas para la configuración externa del metaalmacén de Hive en la clasificación **`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 las referencias secretas: Hive

Example — Especifique las referencias secretas en la configuración externa del metaalmacén de Hive para 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 las referencias secretas para la configuración externa del metaalmacén de Hive en la clasificación **`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"
    }
}
```

Concede acceso a EMR Serverless para recuperar el secreto

Para permitir que EMR Serverless recupere el valor secreto de Secrets Manager, añada la siguiente declaración de política al secreto cuando lo cree. Debe crear su secreto con la KMS clave administrada por el cliente para que EMR Serverless lea el valor secreto. Para obtener más información, consulte los permisos de la clave en el KMS AWS Secrets Manager Guía del usuario.

En la siguiente política, *applicationId* sustitúyalo por el identificador de tu solicitud.

Política de recursos para el secreto

Debe incluir los siguientes permisos en la política de recursos del secreto en AWS Secrets Manager para permitir que EMR Serverless recupere valores secretos. Para garantizar que solo una aplicación específica pueda recuperar este secreto, si lo desea, puede especificar el ID de la aplicación EMR sin servidor como condición en la 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:Región de AWS:aws_account_id:/
applications/applicationId"
        }
      }
    }
  ]
}
```

Cree su secreto con la siguiente política para el gestionado por el cliente AWS Key Management Service (AWS KMS) clave:

Política gestionada por el cliente AWS KMS clave

```
{
    "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.Región de AWS.amazonaws.com"
        }
    }
}
```

Rotando el secreto

La rotación se produce cuando actualizas periódicamente un secreto. Se puede configurar AWS Secrets Manager para rotar automáticamente el secreto según el cronograma que especifique. De esta forma, puede sustituir los secretos a largo plazo por secretos a corto plazo. Esto ayuda a reducir el riesgo de compromiso. EMRServerless recupera el valor secreto de una configuración anotada cuando el trabajo pasa a un estado de ejecución. Si usted o un proceso actualizan el valor secreto en Secrets Manager, deben enviar un nuevo trabajo para que el trabajo pueda recuperar el valor actualizado.

> **Note**
>
> Los trabajos que ya están en ejecución no pueden recuperar un valor secreto actualizado. Esto podría provocar un fallo en el trabajo.

Uso de Amazon S3 Access Grants con EMR Serverless

Descripción general de S3 Access Grants para EMR Serverless

Con las EMR versiones 6.15.0 y posteriores de Amazon, las subvenciones de acceso de Amazon S3 proporcionan una solución de control de acceso escalable que puede utilizar para aumentar el acceso a sus datos de Amazon S3 desde Serverless. EMR Si cuenta con una configuración de permisos compleja o amplia de datos de S3, puede utilizar Access Grants para escalar los permisos de datos de S3 para usuarios, roles y aplicaciones.

Utilice S3 Access Grants para aumentar el acceso a los datos de Amazon S3 más allá de los permisos otorgados por el rol de tiempo de ejecución o los IAM roles asociados a las identidades con acceso a su aplicación EMR sin servidor.

Para obtener más información, consulte Administrar el acceso con S3 Access Grants para Amazon EMR en la Guía de EMR administración de Amazon y Administrar el acceso con S3 Access Grants en la Guía del usuario de Amazon Simple Storage Service.

En esta sección se describe cómo lanzar una aplicación EMR sin servidor que utilice S3 Access Grants para proporcionar acceso a los datos de Amazon S3. Para ver los pasos para usar S3 Access Grants con otras EMR implementaciones de Amazon, consulta la siguiente documentación:

- Uso de S3 Access Grants con Amazon EMR
- Uso de S3 Access Grants con Amazon EMR en EKS

Lance una aplicación EMR sin servidor con S3 Access Grants para la gestión de datos

Puedes activar S3 Access Grants en EMR Serverless e iniciar una aplicación Spark. Cuando su aplicación solicita datos de S3, Amazon S3 brinda credenciales temporales que se limitan al bucket, al prefijo o al objeto.

1. Configura una función de ejecución de tareas para tu aplicación EMR sin servidor. Incluye los IAM permisos necesarios para ejecutar los trabajos de Spark y usar S3 Access Grants `s3:GetDataAccess` y`s3:GetAccessGrantsInstanceForPrefix`:

```
{
    "Effect": "Allow",
```

```
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> Si especificas IAM funciones para la ejecución de tareas que tengan permisos adicionales para acceder directamente a S3, los usuarios podrán acceder a los datos permitidos por la función aunque no tengan el permiso de S3 Access Grants.

2. Lance su aplicación EMR Serverless con una etiqueta de EMR lanzamiento de Amazon de 6.15.0 o superior y la `spark-defaults` clasificación, tal y como se muestra en el siguiente ejemplo. Reemplace los valores en *`red text`* con valores adecuados para su caso 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"
         }
      }]
```

```
}'
```

Consideraciones sobre S3 Access Grants con Serverless EMR

Para obtener información importante sobre soporte, compatibilidad y comportamiento al utilizar Amazon S3 Access Grants con EMR Serverless, consulte Consideraciones sobre S3 Access Grants con Amazon EMR en la Guía de EMR administración de Amazon.

Registro de API llamadas de Amazon EMR Serverless mediante AWS CloudTrail

Amazon EMR Serverless está integrado con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en EMR Serverless. CloudTrail captura todas las API llamadas de EMR Serverless como eventos. Las llamadas capturadas incluyen las llamadas desde la consola EMR sin servidor y las llamadas en código a las operaciones sin EMR servidorAPI. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para EMR Serverless. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a EMR Serverless, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la AWS CloudTrail Guía del usuario.

EMRInformación sin servidor en CloudTrail

CloudTrail está habilitada en su Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en EMR Serverless, esa actividad se registra en un CloudTrail evento junto con otros AWS eventos de servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su Cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos en su Cuenta de AWS, incluidos los eventos de EMR Serverless, crea una ruta. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando crea una ruta en la consola, la ruta se aplica a todos Regiones de AWS. El sendero registra los eventos de todas las regiones del AWS particiona y entrega los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en

los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail servicios e integraciones compatibles
- Configuración de SNS las notificaciones de Amazon para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas las acciones de EMR Serverless se registran CloudTrail y se documentan en la Referencia de EMRServerless API. Por ejemplo, las llamadas a `StartJobRun` y `CancelJobRun` las acciones `CreateApplication` generan entradas en los archivos de CloudTrail registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con root o AWS Identity and Access Management (IAM) credenciales de usuario.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud fue realizada por otra persona AWS servicio.

Para obtener más información, consulte el CloudTrail userIdentity elemento.

Descripción de las entradas de los archivos de registro EMR sin servidor

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las API llamadas públicas, por lo que no aparecen en ningún orden específico.

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `CreateApplication` acción.

```
{
```

```
    "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"
}
```

Validación de conformidad para Amazon EMR Serverless

Auditores externos evalúan la seguridad y el cumplimiento de EMR Serverless como parte de varios AWS programas de cumplimiento, incluidos los siguientes:

- Controles del sistema y la organización (SOC)
- Estándar de seguridad de datos de la industria de tarjetas de pago (PCIDSS)
- Programa federal de gestión de riesgos y autorizaciones (FedRAMP): Moderado
- Ley de Portabilidad y Responsabilidad de los Seguros de Salud () HIPAA

AWS proporciona una lista actualizada con frecuencia de AWS servicios en el ámbito de programas de cumplimiento específicos en AWS Servicios incluidos en el ámbito de aplicación del programa de cumplimiento.

Los informes de auditoría de terceros están disponibles para su descarga mediante AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artefacto.

Para obtener más información acerca de AWS programas de cumplimiento, consulte AWS Programas de cumplimiento.

Su responsabilidad de cumplimiento al utilizar EMR Serverless viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su organización y las leyes y reglamentos aplicables. Si su uso de EMR Serverless está sujeto al cumplimiento de normas como la Fed PCI RAMP ModerateHIPAA, AWS proporciona recursos para ayudar a:

- Guías de inicio rápido sobre seguridad y cumplimiento en las que se analizan las consideraciones arquitectónicas y los pasos para implementar entornos básicos centrados en la seguridad y el cumplimiento en AWS.
- AWS Las guías de cumplimiento para clientes pueden ayudarle a entender el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar Servicios de AWS y mapear la guía con los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST),

el Consejo de Normas de Seguridad de la Industria de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).

- AWS Config se puede utilizar para evaluar en qué medida las configuraciones de los recursos cumplen las prácticas internas, las directrices del sector y las normativas.
- AWS Compliance Resources es una colección de libros de trabajo y guías que pueden aplicarse a su sector y ubicación.
- AWS Security Hub le proporciona una visión completa de su estado de seguridad interno AWS y le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad.
- AWS Audit Manager— esto Servicio de AWS le ayuda a auditar continuamente su AWS uso para simplificar la forma en que gestiona el riesgo y el cumplimiento de las normas y los estándares del sector.

Resiliencia en Amazon EMR Serverless

La AWS la infraestructura global se construye en torno a AWS Regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información acerca de AWS Regiones y zonas de disponibilidad, consulte AWS Infraestructura global.

Además de AWS Amazon EMR Serverless, una infraestructura global, ofrece integración con Amazon S3 EMRFS para ayudarlo a satisfacer sus necesidades de respaldo y resiliencia de datos.

Seguridad de la infraestructura en Amazon EMR Serverless

Como servicio gestionado, Amazon EMR está protegido por AWS seguridad de red global. Para obtener más información AWS servicios de seguridad y cómo AWS protege la infraestructura, consulte AWS Seguridad en la nube. Para diseñar su AWS utilizando las mejores prácticas de seguridad de la infraestructura, consulte el pilar Protección de la infraestructura en la seguridad AWS Un marco bien diseñado.

Usas AWS publicó API llamadas para acceder a Amazon EMR a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Necesitamos TLS 1.2 y recomendamos TLS 1.3.
- Cifre suites con perfecto secreto (PFS), como (Ephemeral Diffie-Hellman) o DHE ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben firmarse con un identificador de clave de acceso y una clave de acceso secreta que esté asociada a un director. IAM O bien, puede utilizar la AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar las solicitudes.

Análisis de configuración y vulnerabilidad en Amazon EMR Serverless

AWS se encarga de las tareas de seguridad básicas, como la aplicación de parches al sistema operativo (SO) huésped y a las bases de datos, la configuración del firewall y la recuperación ante desastres. Estos procedimientos han sido revisados y certificados por los terceros pertinentes. Para obtener más detalles, consulte los siguientes recursos de :

- Validación de conformidad para Amazon EMR Serverless
- Modelo de responsabilidad compartida
- Amazon Web Services: información general de procesos de seguridad (documento técnico)

Puntos finales y cuotas para EMR Serverless

Puntos de conexión de servicio

Para conectarse mediante programación a un Servicio de AWS, se utiliza un punto final. Un punto final es el punto URL de entrada de un AWS servicio web. Además del estándar AWS puntos finales, algunos Servicios de AWS ofrecen FIPS puntos de conexión en regiones seleccionadas. En la siguiente tabla se enumeran los puntos de conexión del servicio Serverless. EMR Para obtener más información, consulte Servicio de AWS puntos finales.

Nombres de las regiones	Región	Punto de conexión	Protocolo
Este de EE. UU. (Ohio)	`us-east-2` (limitado a las siguientes zonas de disponibilidad: `use2-az1use2-az2`, `yuse2-az3`)	`emr-serverless.us-east-2.amazonaws.com`	HTTPS

Este de EE. UU. (Norte de Virginia)	`us-east-1` (limitado a las siguientes zonas de disponibilidad: `use1-az1 use1-az2use1-az4`,`use1-az5`, `yuse1-az6`)	`emr-serverless.us-east-1.amazonaws.com`
		
		`emr-serverless-fips.us-east-1.amazonaws.com`
			HTTPS

Oeste de EE. UU. (Norte de California)	`us-west-1`	`emr-serverless.us-west-1.amazonaws.com`	HTTPS
Oeste de EE. UU. (Oregón)	`us-west-2`	`emr-serverless.us-`	HTTPS

Nombres de las regiones	Región	Punto de conexión	Protocolo

		`west-2.amazonaws.com`
		
		`emr-serverless-fips.us-west-2.amazonaws.com`	

África (Ciudad del Cabo)	`af-south-1`	`emr-serverless.af-south-1.amazonaws.com`	HTTPS
Asia-Pacífico (Hong Kong)	`ap-east-1`	`emr-serverless.ap-east-1.amazonaws.com`	HTTPS
Asia-Pacífico (Yakarta)	`ap-southeast-3`	`emr-serverless.ap-southeast-3.amazonaws.com`	HTTPS
Asia Pacific (Bombay)	`ap-south-1`	`emr-serverless.ap-south-1.amazonaws.com`	HTTPS
Asia-Pacífico (Osaka)	`ap-northeast-3`	`emr-serverless.ap-northeast-3.amazonaws.com`	HTTPS

Nombres de las regiones	Región	Punto de conexión	Protocolo
Asia-Pacífico (Seúl)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
Asia-Pacífico (Singapur)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
Asia-Pacífico (Sídney)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
Asia-Pacífico (Tokio)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS
Canadá (centro)	ca-central-1 (limitado a las siguientes zonas de disponibilidad: cac1-az1 ycac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS
Europe (Fráncfort)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS

Nombres de las regiones	Región	Punto de conexión	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 (Milán)	eu-south-1	emr-serverless.eu-south-1.amazonaws.com	HTTPS
Europa (París)	eu-west-3	emr-serverless.eu-west-3.amazonaws.com	HTTPS
Europa (España)	eu-south-2	emr-serverless.eu-south-2.amazonaws.com	HTTPS
Europa (Estocolmo)	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
Medio Oriente (Baréin)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS

Nombres de las regiones	Región	Punto de conexión	Protocolo
Oriente Medio (UAE)	me-central-1	emr-serverless.me-central-1.amazonaws.com	HTTPS
América del Sur (São Paulo)	sa-east-1	emr-serverless.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (EEUU-Este)	us-gov-east-1	emr-serverless.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (Estados Unidos-Oeste)	us-gov-west-1	emr-serverless.us-gov-west-1.amazonaws.com	HTTPS

Service Quotas

Las cuotas de servicio, también conocidas como límites, son la cantidad máxima de recursos u operaciones de servicio que puede Cuenta de AWS puede usar. EMRServerless recopila las métricas de uso de la cuota de servicio cada minuto y las publica en el `AWS/Usage` espacio de nombres.

> **Note**
>
> New AWS las cuentas pueden tener cuotas iniciales más bajas que pueden aumentar con el tiempo. Amazon EMR Serverless monitorea el uso de las cuentas en cada Región de AWS y, a continuación, aumenta automáticamente las cuotas en función de su uso.

En la siguiente tabla se muestran las cuotas de servicio de EMR Serverless. Para obtener más información, consulte Servicio de AWS cuotas.

Nombre	Límite predeterminado	¿Ajustable?	Descripción
Máximo simultáneo vCPUs por cuenta	16	Sí	El número máximo vCPUs que puede ejecutarse simultáneamente en la cuenta actual Región de AWS.

APIlímites

A continuación se describen los API límites por región de su Cuenta de AWS.

Recurso	Cuota predeterminada
ListApplications	10 transacciones por segundo. Ráfaga de 50 transacciones por segundo.
CreateApplication	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.
DeleteApplication	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.
GetApplication	10 transacciones por segundo. Ráfaga de 50 transacciones por segundo.
UpdateApplication	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.
ListJobRuns	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.

Recurso	Cuota predeterminada
StartJobRun	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.
GetDashboardForJobRun	1 transacción por segundo. Ráfaga de 2 transacciones por segundo.
CancelJobRun	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.
GetJobRun	10 transacciones por segundo. Ráfaga de 50 transacciones por segundo.
StartApplication	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.
StopApplication	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.

Otras consideraciones

La siguiente lista contiene otras consideraciones sobre Serverless. EMR

- EMRServerless está disponible en las siguientes Regiones de AWS:
  - US East (Ohio)
  - Este de EE. UU. (Norte de Virginia)
  - Oeste de EE. UU. (Norte de California)
  - Oeste de EE. UU. (Oregón)
  - África (Ciudad del Cabo)
  - Asia-Pacífico (Hong Kong)
  - Asia-Pacífico (Yakarta)
  - Asia Pacific (Bombay)
  - Asia-Pacífico (Osaka)
  - Asia-Pacífico (Seúl)
  - Asia-Pacífico (Singapur)
  - Asia-Pacífico (Sídney)
  - Asia-Pacífico (Tokio)
  - Canadá (centro)
  - Europa (Fráncfort)
  - Europa (Irlanda)
  - Europa (Londres)
  - Europa (Milán)
  - Europa (París)
  - Europa (España)
  - Europa (Estocolmo)
  - Medio Oriente (Baréin)
  - Oriente Medio () UAE
  - América del Sur (São Paulo)
  - AWS GovCloud (EE. UU.-Este)
  - AWS GovCloud (Estados Unidos-Oeste)

Para obtener una lista de los puntos finales asociados a estas regiones, consulte. Puntos de conexión de servicio

- El tiempo de espera predeterminado para la ejecución de un trabajo es de 12 horas. Puede cambiar esta configuración con la `executionTimeoutMinutes` propiedad de `startJobRun` API o AWS SDK. Puedes ponerlo `executionTimeoutMinutes` en 0 si quieres que tu trabajo no se agote nunca. Por ejemplo, si tiene una aplicación de streaming, puede establecer el valor 0 `executionTimeoutMinutes` para permitir que el trabajo de streaming se ejecute de forma continua.
- La `billedResourceUtilization` propiedad de `getJobRun` API muestra el conjunto vCPU, la memoria y el almacenamiento que AWS ha facturado la ejecución del trabajo. Los recursos facturados incluyen un uso mínimo de 1 minuto para los trabajadores, además de almacenamiento adicional de más de 20 GB por trabajador. Estos recursos no incluyen el uso para trabajadores inactivos y preinicializados.
- Sin VPC conectividad, un trabajo puede acceder a algunos Servicio de AWS puntos finales en el mismo Región de AWS. Estos servicios incluyen Amazon S3, AWS Glue, Amazon CloudWatch Logs, AWS KMS, AWS Security Token Service, Amazon DynamoDB, y AWS Secrets Manager. Puede habilitar la VPC conectividad para acceder a otros Servicios de AWS mediante AWS PrivateLink, pero no estás obligado a hacerlo. Para acceder a servicios externos, puede crear su aplicación con unVPC.
- EMRServerless no es compatibleHDFS. Los discos locales de los trabajadores son un almacenamiento temporal que EMR Serverless utiliza para mezclar y procesar los datos durante la ejecución de los trabajos.
- EMRServerless no es compatible con los existentes. emr-dynamodb-connector

Versiones de lanzamiento de Amazon EMR Serverless

Una EMR versión de Amazon es un conjunto de aplicaciones de código abierto del ecosistema de big data. Cada versión incluye aplicaciones, componentes y características de big data que usted selecciona para que Amazon EMR Serverless implemente y configure cuando ejecute su trabajo.

Con Amazon EMR 6.6.0 y versiones posteriores, puede implementar EMR Serverless. Esta opción de despliegue no está disponible en las EMR versiones anteriores de Amazon. Cuando envíes tu trabajo, debes especificar una de las siguientes versiones compatibles.

Temas

EMR Serverless 7.2.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 7.2.0.

Aplicación	Versión
Apache Spark	3.5.1

Aplicación	Versión
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de la versión 7.2.0 sin servidor

- Lake Formation con EMR Serverless: ahora puede usar AWS Lake Formation para aplicar controles de acceso detallados a las tablas del catálogo de datos respaldadas por S3. Esta capacidad le permite configurar los controles de acceso a nivel de tabla, fila, columna y celda para las consultas de lectura en sus trabajos de EMR Serverless Spark. Para obtener más información, consulte the section called “Lake Formation para FGAC” y the section called “Consideraciones”.

EMR Serverless 7.1.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 7.1.0.

Aplicación	Versión
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.0.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 7.0.0.

Aplicación	Versión
Apache Spark	3.5.0

Aplicación	Versión
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.15.0.

Aplicación	Versión
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de la versión 6.15.0 de Serverless

- TLSsoporte: con las versiones 6.15.0 y posteriores de Amazon EMR Serverless, puedes habilitar la comunicación TLS cifrada mutua entre los trabajadores en tus tareas de Spark. Cuando está habilitada, EMR Serverless genera automáticamente un certificado único para cada trabajador que se aprovisiona en el marco de una ejecución de tareas y que los trabajadores utilizan durante el TLS apretón de manos para autenticarse entre sí y establecer un canal cifrado para procesar los datos de forma segura. Para obtener más información sobre el TLS cifrado mutuo, consulte Cifrado entre trabajadores.

EMR Serverless 6.14.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.14.0.

Aplicación	Versión
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.13.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.13.0.

Aplicación	Versión
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.12.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.12.0.

Aplicación	Versión
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.11.0.

Aplicación	Versión
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de la versión 6.11.0 de Serverless

- Acceda a los recursos de S3 en otras cuentas: con las versiones 6.11.0 y posteriores, puede configurar varios IAM roles para que los asuma al acceder a los buckets de Amazon S3 en diferentes AWS cuentas de Serverless. EMR

EMR Serverless 6.10.0

La siguiente tabla muestra las versiones de la aplicación disponibles con EMR Serverless 6.10.0.

Aplicación	Versión
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de la versión 6.10.0 de Serverless

- Para las aplicaciones EMR sin servidor con la versión 6.10.0 o superior, el valor predeterminado de la propiedad es. `spark.dynamicAllocation.maxExecutors` `infinity` Las versiones anteriores son de forma predeterminada. `100` Para obtener más información, consulte Aumente las propiedades laborales.

EMR Serverless 6.9.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.9.0.

Aplicación	Versión
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRNotas de la versión 6.9.0 de Serverless

- La integración de Amazon Redshift para Apache Spark se incluye en las EMR versiones 6.9.0 y posteriores de Amazon. La integración nativa, que anteriormente era una herramienta de código abierto, es un conector de Spark que puede utilizar para crear aplicaciones de Apache Spark que leen y escriben datos en Amazon Redshift y Amazon Redshift sin servidor. Para obtener más información, consulte Uso de la integración de Amazon Redshift para Apache Spark en Amazon Serverless EMR.
- EMRLa versión 6.9.0 sin servidor añade compatibilidad con AWS La arquitectura Graviton 2 (arm64). Puede usar el `architecture` parámetro para `create-application` y `update-application` APIs para elegir la arquitectura arm64. Para obtener más información, consulte Opciones de arquitectura Amazon EMR Serverless.
- Ahora puede exportar, importar, consultar y unirse a tablas de Amazon DynamoDB directamente desde EMR sus aplicaciones Serverless Spark y Hive. Para obtener más información, consulte Conexión a DynamoDB con Amazon Serverless EMR.

Problemas conocidos

- Si utiliza la integración de Amazon Redshift para Apache Spark y tiene un valor de time, timetz, timestamp o timestamptz con una precisión de microsegundos en formato Parquet, el conector redondea los valores de tiempo al valor de milisegundos más cercano. Como solución alternativa, utilice el parámetro `unload_s3_format` de formato de descarga de texto.

EMR Serverless 6.8.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.8.0.

Aplicación	Versión
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.7.0.

Aplicación	Versión
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

Cambios, mejoras y problemas resueltos específicos del motor

En la siguiente tabla se muestra una nueva función específica del motor.

Cambio	Descripción
Característica	El programador de Tez ahora admite la preferencia sobre la tarea de Tez en lugar de la prioridad sobre el contenedor

EMR Serverless 6.6.0

En la siguiente tabla se enumeran las versiones de la aplicación disponibles con EMR Serverless 6.6.0.

Aplicación	Versión
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMRNotas de la versión inicial sin servidor

- EMRServerless admite la clasificación de configuración de Spark. `spark-defaults` Esta clasificación cambia los valores del `spark-defaults.conf` XML archivo de Spark. Las clasificaciones de configuración le permiten personalizar las aplicaciones. Para obtener más información, consulte Configurar aplicaciones.
- EMRServerless admite las clasificaciones de configuración de Hive`hive-site`,`tez-site`, `emrfs-site` y. `core-site` Esta clasificación puede cambiar los valores del archivo de Hive, el `hive-site.xml` `tez-site.xml` archivo de Tez, la EMRFS configuración EMR de Amazon o el `core-site.xml` archivo de Hadoop, respectivamente. Las clasificaciones de configuración le permiten personalizar las aplicaciones. Para obtener más información, consulte Configurar aplicaciones.

Cambios, mejoras y problemas resueltos específicos del motor

- En la siguiente tabla se muestran los puertos de respaldo de Hive y Tez.

  Cambios en Hive y Tez

Cambio	Descripción
Portabilidad con versiones anteriores	TEZ-4430: Se ha solucionado un problema con la propiedad `tez.task.launch.cmd-opts`

Cambio	Descripción
Portabilidad con versiones anteriores	HIVE-25971: Se corrigieron los retrasos en el cierre de las tareas de Tez debido a un grupo de subprocesos en caché abierto

Historial del documento

En la siguiente tabla se describen los cambios importantes en la documentación desde la última versión de EMR Serverless. Para obtener más información sobre las actualizaciones de esta documentación, puede suscribirse a un RSS feed.

Cambio	Descripción	Fecha
Nueva versión	EMR Serverless 7.2.0	25 de julio de 2024
Nueva versión	EMR Serverless 7.1.0	17 de abril de 2024
Actualización de una política existente.	Se agregó la nueva `Sid` `CloudWatchPolicySt atement` y `EC2Policy Statement` a la mazonEMRServerless ServiceRolePolicy política A.	25 de enero de 2024
Nueva versión	EMR Serverless 7.0.0	29 de diciembre de 2023
Nueva versión	EMR Serverless 6.15.0	17 de noviembre de 2023
Nueva característica	Configure varios IAM roles para que los asuma al acceder a los buckets de Amazon S3 en cuentas diferentes desde EMR Serverless (6.11 y versiones posteriores)	18 de octubre de 2023
Nueva versión	EMR Serverless 6.14.0	17 de octubre de 2023
Nueva característica	Configuración de aplicaciones predeterminada para Serverless EMR	25 de septiembre de 2023
Actualice las propiedades predeterminadas de Hive	Se actualizaron los valores predeterminados de las `hive.driver.disk`	12 de septiembre de 2023

	propiedades de trabajo de `tez.grouping.min-size` Hive `hive.tez.disk.size` `hive.tez.auto.reducer.parallelism`, y.	
Nueva versión	EMR Serverless 6.13.0	11 de septiembre de 2023
Nueva versión	EMR Serverless 6.12.0	21 de julio de 2023
Nueva versión	EMR Serverless 6.11.0	8 de junio de 2023
Actualización de la política de roles vinculados a servicios	Se actualizó el `AmazonEMRServerlessServiceRolePolicy` SLRrol para publicar el uso a nivel de cuenta en el espacio de nombres. `"AWS/Usage"`	20 de abril de 2023
EMR Serverless disponibilidad general (GA)	Esta es la primera versión pública de EMR Serverless.	1 de junio de 2022

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.