Guía del usuario de Amazon EMR sin servidor

Amazon EMR

Amazon EMR: Guía del usuario de Amazon EMR sin servidor

Table of Contents

¿Qué es Amazon EMR sin servidor?

Amazon EMR sin servidor es una opción de implementación para Amazon EMR que proporciona un entorno de tiempo 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 sin servidor, no tiene que configurar, optimizar, proteger ni operar clústeres para ejecutar aplicaciones con estos marcos de trabajo.

EMR sin servidor le ayuda a evitar el aprovisionamiento excesivo o insuficiente de recursos para sus trabajos de procesamiento de datos. EMR sin servidor determina automáticamente los recursos que necesita la aplicación, obtiene estos recursos para procesar sus trabajos y los libera cuando los trabajos 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 preinicializar los recursos que la aplicación necesita cuando crea dicha aplicación.

Con EMR sin servidor, seguirá disfrutando de las ventajas de Amazon EMR, como la compatibilidad con código abierto, la simultaneidad y el rendimiento de tiempo de ejecución optimizado para marcos populares.

EMR sin servidor es adecuado para clientes que desean facilitar el funcionamiento de las aplicaciones 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 sin servidor que aparecen en nuestra Guía del usuario de EMR sin servidor.

Versión de lanzamiento

Una versión de Amazon EMR es un conjunto de aplicaciones de código abierto del ecosistema de macrodatos. Cada versión incluye diferentes aplicaciones, componentes y características de macrodatos que puede seleccionar para que EMR sin servidor los implemente y configure de modo que puedan ejecutar sus aplicaciones. Al crear una aplicación, especifique la versión de lanzamiento. Elija la versión de lanzamiento de Amazon EMR y la versión del marco de código abierto que desee utilizar en su aplicación. Para obtener más información sobre las versiones preliminares, consulte Versiones lanzamiento de Amazon EMR sin servidor.

Aplicación

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

- Elija la versión de lanzamiento de Amazon EMR para la versión del marco de código abierto que desee utilizar. Para determinar la versión de lanzamiento, consulte Versiones lanzamiento de Amazon EMR sin servidor.
- El tiempo de ejecución específico que desea que utilice su aplicación, como Apache Spark o Apache Hive.

Después de crear una aplicación, envíe trabajos de procesamiento de datos o las solicitudes interactivas a la aplicación.

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

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

- Usar diferentes marcos de código abierto
- Usar diferentes versiones de marcos de código abierto para diferentes casos de uso
- Realizar pruebas A/B al actualizar de una versión a otra
- Mantener entornos lógicos separados para los escenarios de prueba y producción
- Proporcionar entornos lógicos separados para los diferentes equipos con controles de costes y seguimiento del uso independientes
- Separar las diferentes aplicaciones de línea de negocio

EMR sin servidor 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 acerca de cómo usar las aplicaciones con EMR sin servidor, consulte Interactuar con una aplicación EMR sin servidor y configurarla.

Ejecución de trabajo

La ejecución de un trabajo es una solicitud enviada a una aplicación EMR sin servidor que la aplicación ejecuta de forma asíncrona y en la que 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 de PySpark que se envía a una aplicación de Apache Spark. Al enviar un trabajo, debe especificar un rol de tiempo de ejecución, creado en IAM, que el trabajo utilice para acceder a los recursos de AWS, como los objetos de Amazon S3. Puede enviar varias solicitudes de ejecución de tareas a una aplicación y cada ejecución de tareas puede utilizar un rol de tiempo de ejecución diferente para acceder a los recursos de AWS. Una aplicación EMR sin servidor comienza a ejecutar trabajos tan pronto como los recibe y ejecuta varias solicitudes de trabajo simultáneamente. Para obtener más información acerca de cómo EMR sin servidor 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 lanzamiento de Amazon EMR. Cuando programa la ejecución de un trabajo, anule estos tamaños.

Cuando envía un trabajo, EMR sin servidor calcula los recursos que la aplicación necesita para el trabajo y programa a los trabajadores. EMR sin servidor divide sus cargas de trabajo en tareas, descarga imágenes, aprovisiona y configura a los trabajadores, y los retira del servicio cuando finaliza el trabajo. EMR sin servidor escala o reduce vertical y automáticamente el número de empleados en función de la carga de trabajo y el paralelismo requeridos en cada etapa del trabajo. Este escalado automático elimina la necesidad de calcular el número de empleados que la aplicación necesita para ejecutar sus cargas de trabajo.

Capacidad preinicializada

EMR sin servidor proporciona una característica de capacidad preinicializada que mantiene a los trabajadores inicializados y listos para responder en cuestión de segundos. Esta capacidad crea de manera efectiva un grupo de calentamiento de trabajadores para una aplicación. Para configurar esta característica para cada aplicación, establezca el parámetro `initial-capacity` 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 cuando se trabaja con EMR sin servidor.

EMR Studio

EMR Studio es la consola de usuario para administrar sus aplicaciones EMR sin servidor. Si no existe un EMR Studio en su cuenta cuando creó su primera aplicación EMR sin servidor, crearemos uno automáticamente para usted. Acceda a EMR Studio desde la consola de Amazon EMR o active el acceso federado desde su proveedor de identidades (IdP) a través de IAM o IAM Identity Center. Al hacerlo, los usuarios pueden acceder a Studio y gestionar las aplicaciones EMR sin servidor sin acceso directo a la consola de Amazon EMR. Para obtener más información sobre cómo funcionan las aplicaciones EMR sin servidor con EMR Studio, consulte Creación de una aplicación EMR sin servidor desde la consola de EMR Studio y Ejecución de trabajos desde la consola de EMR Studio.

Requisitos previos para comenzar a usar EMR sin servidor.

En esta sección se describen los requisitos administrativos previos para ejecutar EMR sin servidor. Estos incluyen la configuración de la cuenta y la administración de permisos.

Temas

Cómo crear una Cuenta de AWS

Si no dispone de una Cuenta de AWS, siga estos pasos para crear una.

Procedimiento para registrarse en Cuenta de AWS

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

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

   Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa 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 le enviará un correo electrónico de confirmación cuando complete el proceso de registro. Puede ver la actividad de la cuenta y administrarla en cualquier momento entrando en https://aws.amazon.com/ y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de registrarse para obtener una Cuenta de AWS, proteja su Usuario raíz de la cuenta de AWS, habilite AWS IAM Identity Center y cree un usuario administrativo para no usar el usuario raíz en las tareas cotidianas.

Protección de Usuario raíz de la cuenta de AWS

1. Inicie sesión en Consola de administración de AWS como propietario de la cuenta; para ello, elija Usuario raíz e introduzca el correo electrónico de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In.
2. Active la autenticación multifactor (MFA) para el usuario raíz.

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

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center.
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre cómo usar Directorio de IAM Identity Center como origen de identidad, consulte Configuración del acceso de los usuarios con el Directorio de IAM Identity Center predeterminado en la Guía del usuario de AWS IAM Identity Center.

Inicio de sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, use la URL de inicio de sesión que se envió a la 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 Inicio de sesión en el portal de acceso de AWS en la Guía del usuario de AWS Sign-In.

Concesión de acceso a usuarios adicionales

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

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center.
2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center.

Concesión de permisos

En entornos de producción, le sugerimos que utilice políticas más específicas. Para ver ejemplos de tales políticas, consulte Ejemplos de políticas de acceso de usuario para EMR sin servidor. Para obtener más información sobre la administración de acceso, consulte Recursos de AWS para administración de acceso en la Guía del usuario de IAM.

Los usuarios que necesitan empezar a usar EMR sin servidor en un entorno de prueba deberán usar una política similar a la siguiente:

JSON

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

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

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 Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center.

- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Concesión de acceso programático

Los usuarios necesitan acceso programático si desean interactuar con AWS fuera de la Consola de administración de AWS. La forma de conceder el acceso programático depende del tipo de usuario que acceda a AWS.

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

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Utiliza credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK y las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: • Para utilizar la AWS CLI, consulte Configuring the AWS CLI to usa AWS IAM Identity Center en la Guía del usuario de AWS Command Line Interface. • Para usar AWS SDK, las herramientas y las API de AWS, consulta IAM Identity

¿Qué usuario necesita acceso programático?	Para	Mediante
		Center authentication en la Guía de referencia del SDK y las herramientas de AWS.
IAM	Utiliza credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK y las API de AWS.	Siguiendo las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del usuario de IAM.
IAM	(No recomendado) Utilizar credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para ver los AWS SDK y las herramientas, consulte Autenticar mediante credenciales a largo plazo en la Guía de referencia de AWS SDK y herramientas. • Para las API de AWS, consulta Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Instalación y configuración de la AWS CLI

Si desea utilizar las API de EMR sin servidor, instale la última versión de AWS Command Line Interface (AWS CLI). No necesita la AWS CLI para utilizar EMR sin servidor desde la consola de EMR Studio y puede empezar sin la CLI siguiendo los pasos que se indican en Introducción a EMR sin servidor con la consola.

Configuración de la AWS CLI

1. Para instalar la versión más reciente de la AWS CLI para macOS, Linux o Windows, consulte Instalación o actualización de la versión más reciente de AWS CLI.
2. Para configurar la AWS CLI y asegurar la configuración de su acceso a los Servicios de AWS, incluido EMR sin servidor, consulte Configuración rápida con `aws configure`.
3. Para verificar la configuración, ingrese el siguiente comando DataBrew en el símbolo del sistema.

   ```
   aws emr-serverless help
   ```

   Los comandos de la AWS CLI utilizan la Región de AWS predeterminada de su configuración, a menos que lo configure con un parámetro o un perfil. Para configurar su Región de AWS con un parámetro, añada el parámetro `--region` a cada comando.

   Para configurar su Región de AWS con un perfil, añada primero un perfil denominado en el archivo `~/.aws/config` o en el archivo `%UserProfile%/.aws/config` (para Microsoft Windows). Siga los pasos en Perfiles denominados para la AWS CLI. A continuación, defina la Región de AWS y otros ajustes con un comando similar al del ejemplo siguiente.

   ```
   [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

La mayoría de los temas orientados a las consolas de esta sección comienzan desde la consola de Amazon EMR. Si aún no ha iniciado sesión en su Cuenta de AWS, inicie sesión, abra la consola de Amazon EMR y continúe con la siguiente sección para empezar a utilizar Amazon EMR.

Introducción a Amazon EMR sin servidor

Este tutorial le ayudará a empezar a utilizar EMR sin servidor al implementar una carga de trabajo de ejemplo de Spark o Hive. Creará, ejecutará y depurará su propia aplicación. En la mayoría de los apartados de este tutorial, le mostraremos opciones predeterminadas.

Antes de iniciar una aplicación de EMR sin servidor, realice las siguientes tareas.

Temas

Otorgar permisos para usar EMR sin servidor

Para usar EMR sin servidor, necesita un rol de usuario o de IAM con una política adjunta que otorgue permisos para EMR sin servidor. Para crear un usuario y asociar la política correspondiente a ese usuario, siga las instrucciones de Concesión de permisos.

Preparar el almacenamiento para EMR sin servidor

En este tutorial, utilizará un bucket de S3 para almacenar los archivos de salida y los registros de la carga de trabajo de muestra de Spark o Hive que ejecutará con una aplicación EMR sin servidor. Para crear un bucket, siga las instrucciones en Crear un bucket en la Guía del usuario de la consola de Amazon Simple Storage Service. Sustituya cualquier referencia adicional a *amzn-s3-demo-bucket* por el nombre del bucket recién creado.

Crear un EMR Studio para ejecutar cargas de trabajo interactivas

Si quiere usar EMR sin servidor para ejecutar consultas interactivas a través de cuadernos alojados en EMR Studio, debe especificar un bucket de S3 y el rol de servicio mínimo para que EMR sin servidor cree un espacio de trabajo. Para ver los pasos de configuración, consulte Configurar un

EMR Studio en la Guía de administración de Amazon EMR. Para más información sobre cargas de trabajo interactivas, consulte Ejecutar cargas de trabajo interactivas con EMR sin servidor a través de EMR Studio.

Crear un rol de tiempo de ejecución del trabajo

Las ejecuciones de trabajos en EMR sin servidor utilizan un rol de tiempo de ejecución que proporciona permisos granulares a Servicios de AWS y recursos específicos en tiempo de ejecución. En este tutorial, un bucket público de S3 aloja los datos y los scripts. El bucket *amzn-s3-demo-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 sin servidor pueda usar el nuevo rol. A continuación, adjunte la política de acceso de S3 requerida a ese rol. Los pasos siguientes le guiarán a través del proceso.

Console

1. Vaya a la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, elija Políticas.
3. Elija Crear política.
4. La página Crear política se abre en una pestaña nueva. Seleccione el Editor de políticas como archivo Json y péguelo a continuación.

> **Important**
>
> Sustituya *amzn-s3-demo-bucket* en la política que aparece a continuación por el nombre real del bucket creado enPreparar el almacenamiento para EMR sin servidor. Se trata de una política básica para el acceso a S3. Para ver más ejemplos de roles de ejecución de trabajos, consulte Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.

JSON

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

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

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

5. Seleccione Siguiente para ingresar un nombre para la política, como `EMRServerlessS3AndGlueAccessPolicy`, y luego Crear política.
6. En el panel de navegación izquierdo de la consola de IAM, elija Roles.
7. Seleccione Crear rol.
8. Para tipo de rol, elija Política de confianza personalizada e introduzca la siguiente política de confianza. Esto permite que los trabajos enviados a sus aplicaciones Amazon EMR sin servidor accedan a otros Servicios de AWS en su nombre.

   JSON

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

9. Seleccione Siguiente para ir a la página Añadir permisos y, a continuación, elija EMRServerlessS3AndGlueAccessPolicy.
10. En la página Nombrar, revisar y crear, para el Nombre de rol, introduzca un nombre para su rol, por ejemplo, `EMRServerlessS3RuntimeRole`. Para crear este nuevo rol de IAM, elija Crear un rol.

CLI

1. Cree un archivo con el nombre `emr-serverless-trust-policy.json`, que contenga la política de confianza que se va a utilizar para el rol de IAM. El archivo debe contener la política siguiente.

   JSON

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

2. Creación de un rol de IAM denominado `EMRServerlessS3RuntimeRole`. Use 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
   ```

   Anote el ARN en el resultado. Se utiliza el ARN del nuevo rol durante el envío del trabajo, que en adelante se denomina *`job-role-arn`*.

3. Cree un archivo con un nombre `emr-sample-access-policy.json` que defina la política de IAM para su carga de trabajo. Esto proporciona acceso de lectura al script y a los datos almacenados en buckets públicos de S3 y acceso de lectura y escritura a *`amzn-s3-demo-bucket`*.

> **Important**
>
> Sustituya *amzn-s3-demo-bucket* en la política que aparece a continuación por el nombre real del bucket creado en Preparar el almacenamiento para EMR sin servidor. Se trata de una política básica para el acceso a Glue y S3 de AWS. Para ver más ejemplos de roles de ejecución de trabajos, consulte Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.

JSON

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

```
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

4. Cree una política de IAM denominada `EMRServerlessS3AndGlueAccessPolicy` con el archivo de políticas que creó en el Paso 3. Tome nota del ARN en el resultado, ya que utilizará el ARN de la nueva política en el siguiente paso.

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

   Anote el ARN de la nueva política en el resultado. Lo sustituirá *`policy-arn`* en el siguiente paso.

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

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

Introducción a EMR sin servidor con la consola

En esta sección se describe cómo trabajar con EMR sin servidor, incluida la creación de un EMR Studio. También describe cómo enviar las ejecuciones de trabajos y cómo ver los registros.

Pasos que completar

- Paso 1: cree una aplicación de EMR sin servidor
- Paso 2: envíe una ejecución de trabajo o una carga de trabajo interactiva
- Paso 3: visualice la IU y los registros de la aplicación
- Paso 4: Limpiar

Paso 1: cree una aplicación de EMR sin servidor

Cree una nueva aplicación con EMR sin servidor de la siguiente manera.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EMR en https://console.aws.amazon.com/emr.
2. En el panel de navegación izquierdo, elija EMR sin servidor para ir a la página de inicio de EMR sin servidor.
3. Para crear o administrar aplicaciones EMR sin servidor, necesita la IU de EMR Studio.

   - Si ya tiene un EMR Studio en la Región de AWS donde desea crear una aplicación y, a continuación, seleccione Administrar aplicaciones para ir a su EMR Studio, o seleccione el estudio que desee utilizar.
   - Si no tiene un EMR Studio en la Región de AWS donde desea crear una aplicación, elija Cómo comenzar y, a continuación, elija Crear e iniciar Studio. EMR sin servidor crea un EMR Studio para que pueda crear y administrar aplicaciones.
4. En la IU de Crear Studio que se abre en una nueva pestaña, introduzca 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, seleccione 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 la configuración más tarde.

   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 una ejecución de trabajo o una carga de trabajo interactiva para enviar una ejecución de trabajo o una carga de trabajo interactiva.

Paso 2: envíe una ejecución de trabajo o una carga de trabajo interactiva

Spark job run

En este tutorial, utilizamos un script de PySpark para calcular el número de apariciones de 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. Cargue el script de ejemplo `wordcount.py` en su nuevo bucket con el siguiente comando.

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

2. Al completar Paso 1: cree una aplicación de EMR sin servidor, accederá a la página de los Detalles de la aplicación en EMR Studio. Allí, elija la opción Enviar trabajo.
3. En la página Enviar trabajo, complete lo siguiente.
   - En el campo Nombre, escriba el nombre con el que desee llamar a la ejecución de trabajo.
   - En el campo Rol de tiempo de ejecución, escriba el nombre del rol que ha creado en Crear un rol de tiempo de ejecución del trabajo.
   - En el campo Ubicación del script, escriba `s3://amzn-s3-demo-bucket/scripts/wordcount.py` como el URI de S3.
   - En el campo Argumentos del script, escriba `["s3://amzn-s3-demo-bucket/emr-serverless-spark/output"]`.
   - En la sección Propiedades de Spark, seleccione Editar como texto e introduzca 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 de trabajo, elija Enviar trabajo .
5. En la pestaña Ejecuciones de trabajos, 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. Cargue `hive-query.ql` en el bucket de S3 con el comando siguiente.

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

3. Al completar Paso 1: cree una aplicación de EMR sin servidor, accederá a la página de los Detalles de la aplicación en EMR Studio. Allí, elija la opción Enviar trabajo.
4. En la página Enviar trabajo, complete lo siguiente.

   - En el campo Nombre, escriba el nombre con el que desee llamar a la ejecución de trabajo.
   - En el campo Rol de tiempo de ejecución, escriba el nombre del rol que ha creado en Crear un rol de tiempo de ejecución del trabajo.
   - En el campo Ubicación del script, escriba `s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql` como el URI de S3.
   - En la sección Propiedades de Hive, 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 el siguiente JSON.

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

5. Para iniciar la ejecución de trabajo, elija Enviar trabajo.
6. En la pestaña Ejecuciones de trabajos, debería ver el nuevo trabajo ejecutándose con el estado En ejecución.

Interactive workload

Con Amazon EMR 6.14.0 y versiones posteriores, puede usar cuadernos alojados en EMR Studio para ejecutar cargas de trabajo interactivas para Spark en EMR sin servidor. Para obtener más información, incluidos los permisos y requisitos previos, consulte Ejecutar cargas de trabajo interactivas con EMR sin servidor a través de EMR Studio.

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

1. Vaya 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 banner de la parte superior de la pantalla.
2. Para acceder a un cuaderno, seleccione un espacio de trabajo o cree un nuevo espacio de trabajo. Use el Inicio rápido para abrir tu espacio de trabajo en una pestaña nueva.
3. Vaya a la pestaña recién abierta. Seleccione el icono Computar en el panel de navegación izquierdo. Seleccione EMR sin servidor como el Tipo de computación.
4. Seleccione la aplicación con capacidad interactiva que ha creado en la sección anterior.

5. En el campo Rol de ejecución, introduzca el nombre del rol de IAM que la aplicación EMR sin servidor puede asumir para la ejecución del trabajo. Para obtener más información sobre los roles de tiempo de ejecución, consulte Roles de tiempo de ejecución de trabajos en la Guía del usuario de Amazon EMR sin servidor.
6. Seleccione Adjuntar. Esto puede llevar un minuto en completarse. La página se actualizará cuando se adjunte.
7. Escoja un kernel e inicie un cuaderno. También puede buscar cuadernos de ejemplo en EMR sin servidor y copiarlos en su espacio de trabajo. Para acceder a los cuadernos de ejemplo, vaya hasta el **{...}** menú de navegación de la izquierda y explore los cuadernos que tengan `serverless` en el nombre del archivo del cuaderno.
8. En el cuaderno, puede acceder al enlace del registro de controladores y a un enlace a la IU de Apache Spark, una interfaz en tiempo real que proporciona métricas para monitorizar tu trabajo. Para obtener más información, consulte Monitorización de aplicaciones y trabajos de EMR sin servidor en la Guía del usuario de Amazon EMR sin servidor.

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 puede iniciar previamente la aplicación y tenerla lista antes de adjuntarla al espacio de trabajo.

Paso 3: visualice la IU y los registros de la aplicación

Para ver la IU de la aplicación, primero identifique la ejecución del trabajo. Hay disponible una opción para la IU de Spark o la IU de Hive Tez en la primera fila de opciones para la ejecución de ese trabajo, según el tipo de trabajo. Seleccione la opción apropiada.

Si ha elegido la IU de Spark, seleccione la pestaña Ejecutores para ver los registros de los controladores y ejecutores. Si ha elegido la IU de Hive Tez, seleccione la pestaña Todas las tareas para ver los registros.

Una vez que el estado de ejecución del trabajo se muestra como Correcta, podrá ver el resultado del trabajo 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 Enumerar aplicaciones. Seleccione la aplicación que ha creado y elija Acciones → Detener para detener la aplicación. Cuando la aplicación esté en el estado `STOPPED`, seleccione la misma aplicación y elija Acciones → Eliminar.

Para ver más ejemplos de cómo ejecutar trabajos de Spark y Hive, consulte Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor y Uso de configuraciones de Hive al ejecutar trabajos de EMR sin servidor.

Introducción a la AWS CLI

Comience a utilizar EMR sin servidor desde la AWS CLI con comandos para crear una aplicación, ejecutar trabajos, comprobar el resultado de la ejecución de trabajos y eliminar sus recursos.

Paso 1: cree una aplicación de EMR sin servidor

Use el comando de `emr-serverless create-application` para crear su primera aplicación EMR sin servidor. Debe especificar el tipo de aplicación y la etiqueta de versión de Amazon EMR asociada a la versión de la aplicación que desee utilizar. El nombre de la aplicación es opcional.

Spark

Para crear una aplicación de Spark, use el siguiente comando.

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

Hive

Para crear una aplicación de Hive, use el siguiente comando.

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

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

Antes de continuar con Paso 2: envíe una ejecución de trabajo a su aplicación EMR sin servidor, asegúrese de que su aplicación haya alcanzado el estado `CREATED` con la API de `get-application`.

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

EMR sin servidor crea trabajadores para adaptarse a los trabajos solicitados. De forma predeterminada, se crean bajo demanda, pero también puede especificar una capacidad preinicializada estableciendo el parámetro `initialCapacity` al crear la aplicación. También puede limitar la capacidad máxima total que puede utilizar una aplicación con el parámetro `maximumCapacity`. Para obtener más información sobre estas opciones, consulte Configuración de una aplicación cuando se trabaja con EMR sin servidor.

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

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

Spark

En este paso, utilizamos un script de PySpark para calcular el número de apariciones de 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 tiempo de ejecución de Spark a los directorios `/output` y `/logs` del bucket de S3 que creó.

Para ejecutar un trabajo de Spark

1. Use el siguiente comando para copiar el script de ejemplo que ejecutaremos en su nuevo bucket.

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

2. En el siguiente comando, sustituya *application-id* por el ID de la aplicación. Sustituya *job-role-arn* por el ARN del rol del tiempo de ejecución en el que creó Crear un rol de tiempo de ejecución del trabajo. Sustituya *job-run-name* por el nombre con el que desee llamar a su ejecución de trabajos. Sustituya todas cadenas *amzn-s3-demo-bucket* por el bucket de Amazon S3 que creó y añada el `/output` a la ruta. Esto crea una nueva carpeta en su bucket donde EMR sin servidor puede copiar los archivos de salida de la 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://amzn-s3-demo-bucket/scripts/wordcount.py",
          "entryPointArguments": ["s3://amzn-s3-demo-bucket/emr-serverless-
spark/output"],
          "sparkSubmitParameters": "--conf spark.executor.cores=1
 --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
 spark.driver.memory=4g --conf spark.executor.instances=1"
        }
    }'
```

3. Anote el ID de ejecución del trabajo que se devuelve como resultado. Sustituya *job-run-id* por este ID en los siguientes pasos.

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 cuando inicie 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. Cargue `hive-query.ql` en el bucket de S3 con el comando siguiente.

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

3. En el siguiente comando, sustituya *application-id* por su propio ID de aplicación. Sustituya *job-role-arn* por el ARN del rol del tiempo de ejecución en el que creó Crear un rol de tiempo de ejecución del trabajo. Sustituya todas las cadenas *amzn-s3-demo-bucket* por el bucket de Amazon S3 que creó y añada `/output` y `/logs` a la ruta. Esto crea nuevas carpetas en su bucket, donde EMR sin servidor puede copiar los archivos de registro y de salida de su aplicación.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
          "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql",
          "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
      "applicationConfiguration": [{
        "classification": "hive-site",
          "properties": {
            "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/hive/scratch",
            "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/hive/warehouse",
            "hive.driver.cores": "2",
            "hive.driver.memory": "4g",
            "hive.tez.container.size": "4096",
            "hive.tez.cpu.vcores": "1"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/emr-serverless-hive/logs"
           }
        }
    }'
```

4. Anote el ID de ejecución del trabajo que se devuelve como resultado. Sustituya *job-run-id* por este ID en los siguientes pasos.

Paso 3: revise el resultado de la ejecución del trabajo

La ejecución del trabajo tarda normalmente de 3 a 5 minutos en completarse.

Spark

Puede comprobar el estado de su 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://amzn-s3-demo-bucket/emr-serverless-spark/logs`, puede encontrar los registros de este trabajo específico en el que se está ejecutando `s3://amzn-s3-demo-bucket/emr-serverless-spark/logs/applications/application-id/jobs/job-run-id`.

En el caso de las aplicaciones Spark, EMR sin servidor envía los registros de eventos cada 30 segundos a la carpeta `sparklogs` en el destino del registro de S3. Cuando termine su 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 un `driver` o un `executor`. El resultado del trabajo de PySpark se carga en `s3://amzn-s3-demo-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://amzn-s3-demo-bucket/emr-serverless-hive/logs`, puede encontrar los registros de este trabajo específico en el que se está ejecutando `s3://amzn-s3-demo-bucket/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id`.

Para las aplicaciones de Hive, EMR sin servidor carga continuamente el controlador Hive en la carpeta `HIVE_DRIVER`, y los registros de los trabajos de Tez en la carpeta `TEZ_TASK` del destino de registro de S3. Cuando la ejecución del trabajo alcance el estado`SUCCEEDED`, el resultado de la consulta de Hive estará disponible en la ubicación de Amazon S3 que especificó en el campo `monitoringConfiguration` de `configurationOverrides`.

Paso 4: Limpiar

Cuando haya terminado de trabajar con este tutorial, considere eliminar los recursos que creó. Le recomendamos que libere recursos que no tenga intención de volver a utilizar.

Elimine la aplicación

Para eliminar la aplicación, utilice el comando siguiente.

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

Elimine su bucket de registro S3

Para eliminar su bucket de registro y salida de S3, utilice el siguiente comando. Sustituya *amzn-s3-demo-bucket* por el nombre real del bucket de S3 que creó en Preparar el almacenamiento para EMR sin servidor.

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

Elimine el rol de tiempo de ejecución de su trabajo

Para eliminar el rol de tiempo de ejecución, desasocie la política del rol. Entonces, puede eliminar tanto el rol como la política.

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

Para eliminar el rol, utilice el comando siguiente.

```
aws iam delete-role \
```

```
    --role-name EMRServerlessS3RuntimeRole
```

Para eliminar la política que se asoció al rol, utilice el siguiente comando.

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

Para ver más ejemplos de cómo ejecutar trabajos de Spark y Hive, consulte Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor y Uso de configuraciones de Hive al ejecutar trabajos de EMR sin servidor.

Interactuar con una aplicación EMR sin servidor y configurarla

En esta sección se explica cómo interactuar con la aplicación de Amazon EMR sin servidor con la AWS CLI. También describe la configuración de una aplicación, la realización de personalizaciones y los valores predeterminados de los motores Spark y Hive.

Temas

Estados de la aplicación

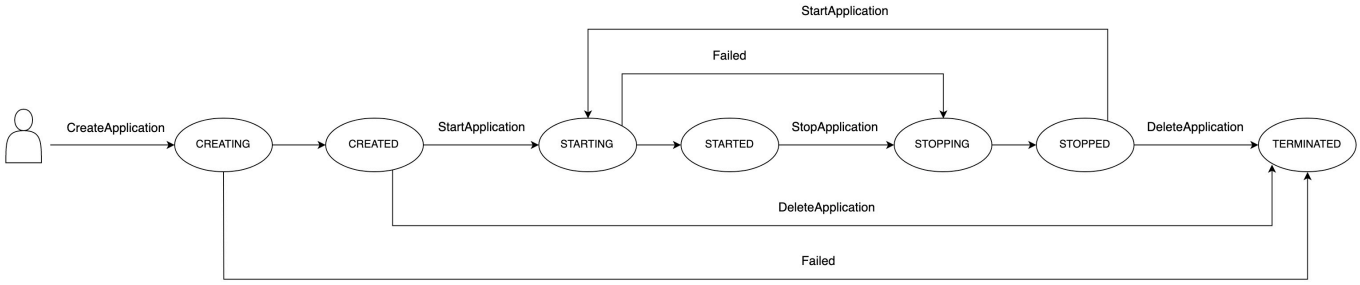
Al crear una aplicación con EMR sin servidor, la ejecución de la aplicación pasa al estado `CREATING`. A continuación, pasa por los estados siguientes hasta que termina de ejecutarse correctamente (finaliza con el código `0`) o no (finaliza con un código distinto de cero).

Las aplicaciones pueden tener los siguientes estados:

Estado	Descripción
Creación	La aplicación se está preparando y aún no está lista para usarse.
Creación	La aplicación se ha creado pero aún no ha aprovisionado capacidad. Puede modificar la

Estado	Descripción
	aplicación para cambiar su configuración de capacidad inicial.
Inicio	La aplicación se está iniciando y aprovisio nando capacidad.
Iniciada	La aplicación está lista para aceptar nuevos trabajos. La aplicación solo acepta trabajos cuando se encuentra en este estado.
Deteniéndose	Se han completado todos los trabajos y la aplicación está agotando su capacidad.
Detenida	La aplicación está detenida y no hay recursos en ejecución en la aplicación. Puede modificar la aplicación para cambiar su configuración de capacidad inicial.
Finalizada	La aplicación ha finalizado y no aparece en su lista de aplicaciones.

En el siguiente diagrama, se ilustra la trayectoria de los estados de las aplicaciones EMR sin servidor.



Creación de una aplicación EMR sin servidor desde la consola de EMR Studio

Cree, administre y acceda a aplicaciones EMR sin servidor desde la consola de EMR Studio Para ir a la consola de EMR Studio, siga las instrucciones de Introducción a la consola.

Crear una aplicación

Con la página Crear aplicación, cree una aplicación EMR sin servidor siguiendo estos pasos.

1. En el campo Nombre, escriba el nombre con el que desee llamar a la aplicación.
2. En el campo Tipo, elija Spark o Hive como el tipo de aplicación.
3. En el campo Versión de lanzamiento, elija el número de versión de EMR.
4. En las opciones de Arquitectura, elija la arquitectura del conjunto de instrucciones que utilizará. Para obtener más información, consulta Opciones de la arquitectura de Amazon EMR sin servidor.

   - arm64: arquitectura ARM de 64 bits, para usar procesadores Graviton
   - x86_64: arquitectura x86 de 64 bits, para usar procesadores basados en x86
5. Hay dos opciones de configuración de la aplicación para los campos restantes: la configuración predeterminada y la 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 sus VPC. 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 controladores 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: el comportamiento de inicio y parada automáticos de la aplicación.
   - Conexiones de red: conectividad de red a los recursos de la VPC.

- 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 cuando se trabaja con EMR sin servidor Para obtener más información acerca de la configuración de red, consulte Configuración del acceso a la VPC para que las aplicaciones EMR sin servidor se conecten a los datos.

6. Para crear la aplicación, elija Crear aplicación.

Enumerar las aplicaciones de la consola de EMR Studio

Puede acceder a todas las aplicaciones EMR sin servidor existentes en la página Enumerar aplicaciones. Puede elegir el nombre de una aplicación para ir a la página de Detalles de esa aplicación.

Gestionar las aplicaciones desde la consola de EMR Studio

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

Iniciar la aplicación

Seleccione esta opción para iniciar manualmente una aplicación.

Detener 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, consulte Estados 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 la versión de una aplicación para actualizarla a una versión diferente de Amazon EMR, 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, consulte Crear 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, consulte Estados de la aplicación.

Interacción con su aplicación EMR sin servidor en la AWS CLI

Cree, describa y elimine aplicaciones individuales desde la AWS CLI. También puede enumerar todas sus aplicaciones para acceder a estas de un vistazo. En esta sección se describe cómo realizar estas acciones. Para ver más operaciones de la aplicación, como iniciar, detener y actualizar la aplicación, consulte la Referencia de la API de EMR sin servidor. Para ver ejemplos de cómo usar la API de EMR sin servidor mediante AWS SDK para Java, consulte los ejemplos de Java en nuestro repositorio de GitHub. Para ver ejemplos de cómo usar la API de EMR sin servidor mediante AWS SDK for Python (Boto), consulte los ejemplos de Python en nuestro repositorio de GitHub.

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

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

Para enumerar todas las aplicaciones, llame a las `list-applications`. Este comando devuelve las mismas propiedades como `get-application`, pero las incluye todas sus aplicaciones.

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

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

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

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

Configuración de una aplicación cuando se trabaja con EMR sin servidor

Configure las aplicaciones que utiliza con EMR sin servidor. Por ejemplo, establezca la capacidad máxima a la que puede escalar verticalmente una aplicación, configurar la capacidad preinicializada a fin de que los controladores y los trabajadores estén preparados para responder y especificar un conjunto común de configuraciones de tiempo de ejecución y monitorización a nivel de aplicación. En las siguientes páginas se describe cómo configurar las aplicaciones cuando se utiliza EMR sin servidor.

Temas

Descripción del comportamiento de las aplicaciones en EMR sin servidor

En esta sección se describe el comportamiento de envío de trabajos, la configuración de la capacidad para el escalado y los ajustes de configuración del trabajador para EMR sin servidor.

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 característica.

Parada automática: una aplicación está configurada de forma predeterminada para que se detenga automáticamente cuando esté inactiva durante 15 minutos. Cuando una aplicación cambia al estado `STOPPED`, 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 característica.

Maximum capacity (Capacidad máxima)

Puede configurar la capacidad máxima hasta la que puede escalarse verticalmente 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 a los tamaños de los trabajadores admitidos multiplicando el número de trabajadores por sus tamaños. Por ejemplo, si desea limitar la aplicación a 50 trabajadores con 2 vCPU, 16 GB de memoria y 20 GB de disco, establezca la capacidad máxima en 100 vCPU, 800 GB de memoria y 1000 GB de disco.

Configuraciones de trabajadores admitidas

La siguiente tabla muestra las configuraciones y los tamaños de trabajadores admitidos que se pueden especificar para EMR sin servidor. Configure diferentes tamaños para los controladores y los ejecutores en función de las necesidades de su carga de trabajo.

Configuraciones y tamaños de trabajadores

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

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 elije 4 vCPU para su instancia de trabajador y una capacidad de almacenamiento preinicializada de 30 GB, establezca un valor de aproximadamente 27 GB como memoria ejecutora para su trabajo de Spark. Esto maximiza el uso de la capacidad preinicializada. La memoria utilizable es 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 uno máximo de 200 GB. Solo paga por el almacenamiento adicional de más de 20 GB que configure por trabajador.

Capacidad preinicializada para trabajar con una aplicación en EMR sin servidor

EMR sin servidor ofrece una característica opcional que mantiene al controlador y a los trabajadores preinicializados y listos para responder en cuestión de segundos. Esto crea de manera efectiva un grupo de calentamiento de trabajadores para una aplicación. Esta característica se denomina capacidad preinicializada. Para configurar esta característica, establezca el parámetro `initialCapacity` de una aplicación en función del número de trabajadores que desee preinicializar. Con la capacidad de trabajadores preinicializada, los trabajos comienzan inmediatamente. Esto resulta ideal cuando se desean implementar aplicaciones iterativas y trabajos urgentes.

La capacidad preinicializada permite que un grupo de calentamiento de trabajadores esté listo para iniciar los trabajos y las sesiones en cuestión de segundos. Deberá pagar por los trabajadores aprovisionados y preinicializados incluso cuando la aplicación esté inactiva, por lo que le sugerimos habilitarla para casos de uso que se beneficien del rápido tiempo de inicio, así como dimensionarla para un uso óptimo de los recursos. Las aplicaciones EMR sin servidor se apagan automáticamente cuando están inactivas. Sugerimos mantener esta característica activada cuando se utilicen trabajadores preinicializados para evitar cargos inesperados.

Al enviar un trabajo, si hay trabajadores de `initialCapacity` 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 de `initialCapacity`, la aplicación solicita y obtiene trabajadores adicionales, hasta el límite máximo de recursos establecido para la aplicación. Cuando un trabajo termina de ejecutarse, libera a los trabajadores que utilizó y el número de recursos disponibles para la aplicación vuelve a la `initialCapacity`. Una aplicación mantiene la `initialCapacity` de recursos incluso después de que los trabajos terminen de ejecutarse. La aplicación libera los recursos sobrantes más allá de la `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 queda inactiva cuando la aplicación se detiene. Una aplicación pasa a ese estado `STARTED` solo si la capacidad preinicializada solicitada se ha creado y está lista para usarse. Durante todo el tiempo que la aplicación esté en el estado `STARTED`, EMR sin servidor mantiene la capacidad preinicializada disponible para ser utilizada o en uso por trabajos o cargas de trabajo interactivas. La característica restaura la capacidad de los contenedores liberados o con errores. Esto mantiene el número de trabajadores que especifica el parámetro `InitialCapacity`. El estado de una aplicación sin capacidad preinicializada puede cambiar inmediatamente de `CREATED` a `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 aplicación detenida se inicia automáticamente cuando envía un nuevo trabajo. Puede establecer estas configuraciones de inicio y detención automáticas cuando cree la aplicación o cambiarlas cuando la aplicación esté en un estado `CREATED` o `STOPPED`.

Puede cambiar los recuentos de la `InitialCapacity` y especificar configuraciones de computación, como CPU, memoria y disco, para cada trabajador. Como no puede realizar modificaciones parciales, debe especificar todas las configuraciones de computación al cambiar los valores. Solo puede cambiar las configuraciones cuando la aplicación esté en el estado `CREATED` o `STOPPED`.

> **Note**
>
> Para optimizar el uso de los recursos de su aplicación, le sugerimos alinear los tamaños de los contenedores con los tamaños de los trabajadores con capacidad preinicializados. Por ejemplo, si configura el tamaño de tu ejecutor de Spark en 2 CPU y tu memoria en 8 GB, pero el tamaño de su trabajador con capacidad preinicializado es de 4 CPU 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 macrodatos específicos. Por ejemplo, cuando una carga de trabajo se ejecute en Apache Spark, especifique cuántos trabajadores comienzan como controladores y cuántos comienzan como ejecutores. Del mismo modo, cuando use Apache Hive, especifique cuántos trabajadores comienzan como controladores de Hive y cuántos deben ejecutar las tareas de Tez.

Configuración de una aplicación que ejecute Apache Hive con capacidad preinicializada

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

```
  }'
```

Configuración de una aplicación que ejecute Apache Spark con capacidad preinicializada

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

Puede especificar un conjunto común de configuraciones de tiempo de ejecución y monitorización a nivel de aplicación para todos los trabajos que envía 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 en el envío del trabajo.
- Anule las configuraciones predeterminadas durante la ejecución del trabajo.

En las siguientes secciones se proporcionan más detalles y un ejemplo como contexto adicional.

Declaración de las configuraciones a nivel de aplicación

Puede especificar las propiedades de configuración del registro y del 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 campo `monitoringConfiguration`. Para obtener más información sobre el registro de EMR sin servidor, consulte Almacenamiento de registros.

`runtimeConfiguration`

Para especificar propiedades de configuración en tiempo de ejecución como `spark-defaults`, proporcione un objeto de configuración en el campo `runtimeConfiguration`. 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 configuración de Spark.

Las clasificaciones de configuración disponibles varían en función de la versión específica de EMR sin servidor. Por ejemplo, las clasificaciones para el Log4j personalizado `spark-`

`driver-log4j2` y `spark-executor-log4j2` solo están disponibles en las versiones 6.8.0 y posteriores. Para obtener una lista de propiedades específicas de la aplicación, consulte Propiedades de trabajo de Spark y Propiedades de los trabajos de Hive.

También puede configurar las propiedades de Apache Log4j2, AWS Secrets Managerpara la protección de datos, así como el tiempo de ejecución de Java 17 al nivel de la aplicación.

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

JSON

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

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 Guía del usuario de AWS Secrets Manager.

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

Declaración de ejemplo

En el siguiente ejemplo, se muestra cómo declarar 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://amzn-s3-demo-logging-bucket/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

Anulación de 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 la API de `StartJobRun`. Luego, EMR sin servidor combina las configuraciones que especifique en el nivel de aplicación y en el nivel de trabajo para determinar las configuraciones para la ejecución del trabajo.

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

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

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

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

Al iniciar un trabajo, si no especifica una configuración concreta, esta 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 solicitud `StartJobRun` con sus valores de anulación.
- Añadir una configuración adicional: añada el nuevo parámetro de configuración en la solicitud `StartJobRun` 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 dará error.

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

  - **`cloudWatchLoggingConfiguration`**: para eliminar `cloudWatchLogging`, pase el indicador de habilitación como `false`.
  - **`managedPersistenceMonitoringConfiguration`**: para eliminar la configuración de persistencia administrada y volver al estado activado predeterminado, introduzca una declaración vacía `{}` para la configuración.
  - **`s3MonitoringConfiguration`**: para eliminar `s3MonitoringConfiguration`, pase una declaración vacía `{}` para la configuración.

Anulación de ejemplo

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://amzn-s3-demo-destination-bucket1/
wordcount_output"]
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [
```

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

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 configuración de Spark.

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

Personalización de una imagen de EMR sin servidor

A partir de Amazon EMR 6.9.0, utilice imágenes personalizadas para empaquetar dependencias de aplicaciones y entornos de tiempo de ejecución en un único contenedor con Amazon EMR sin servidor. Esto simplifica la forma en que administra las dependencias de la carga de trabajo y hace que sus paquetes sean más portátiles. Cuando personaliza su imagen de EMR sin servidor, ofrece las siguientes ventajas:

- Instale y configure paquetes que se hayan optimizado para sus cargas de trabajo. Estos paquetes no están ampliamente disponibles en la distribución pública de los entornos de tiempo de ejecución de Amazon EMR.
- Integra EMR sin servidor con los procesos de compilación, prueba e implementación establecidos actualmente en su organización, incluido el desarrollo y las pruebas locales.
- Aplica procesos de seguridad establecidos, como el escaneo de imágenes, que cumplan con los requisitos de cumplimiento y gobernanza de su organización.
- Le permite usar sus propias versiones de JDK y Python para sus aplicaciones.

EMR sin servidor proporciona imágenes que puede utilizar como base cuando crea sus propias imágenes. La imagen base proporciona los jars, la configuración y las bibliotecas esenciales para que la imagen interactúe con EMR sin servidor. Puede encontrar la imagen base en la Galería pública de Amazon ECR. Utilice 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 versión 6.9.0 de Amazon EMR, 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 de EMR sin servidor, complete estos requisitos previos.

1. Cree un repositorio de Amazon ECR en la misma Región de AWS que utiliza para lanzar las aplicaciones EMR sin servidor. Para crear un repositorio de Amazon ECR privado, consulte Creación de un repositorio privado.
2. Para conceder a los usuarios acceso a su repositorio de Amazon ECR, añada las siguientes políticas a los usuarios y roles que crean o actualizan aplicaciones EMR sin servidor con imágenes de este repositorio.

   JSON

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

```
}
```

Para ver más ejemplos de políticas basadas en identidad de Amazon ECR, consulte Amazon Elastic Container Registry identity-based policy examples.

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

En primer lugar, cree un Dockerfile que comience con una instrucción `FROM` que utilice la imagen base que prefiera. Después de la instrucción `FROM`, incluya cualquier modificación que desee realizar en la imagen. La imagen base establece automáticamente el `USER` en `hadoop`. Esta configuración no tiene permisos para todas las modificaciones que incluya. Como solución alternativa, establezca el `USER` en `root`, modifique la imagen y, a continuación, establezca el `USER` de nuevo en `hadoop:hadoop`. Para consultar ejemplos de casos de uso comunes, consulte Uso de imágenes personalizadas con EMR sin servidor.

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

USER root
# MODIFICATIONS GO HERE

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

Una vez tenga el Dockerfile, compile 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: valide la imagen localmente

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

Una vez instalada 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
```

Aparecerá una salida similar a la 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: cargue la imagen en su repositorio de Amazon ECR

Inserte la imagen de Amazon ECR en su repositorio de Amazon ERC con los siguientes comandos. Asegúrese de tener los permisos de IAM correctos para enviar la imagen a su repositorio. Para obtener más información, consulte Insertar una imagen en la Guía del usuario de Amazon ECR.

```
# login to ECR repo
```

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

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

Paso 4: cree o actualice una aplicación con imágenes personalizadas

Elija la pestaña Consola de administración de AWS o la pestaña AWS CLI según la forma en que desee iniciar la aplicación y, a continuación, complete los siguientes pasos.

Console

1. Inicie sesión en la consola de EMR Studio en https://console.aws.amazon.com/emr. Vaya 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 Uso de la imagen personalizada con esta aplicación.
4. Pegue el URI de imagen de Amazon ECR en el campo URI de imagen. EMR sin servidor utiliza esta imagen para todos los tipos de trabajadores de la aplicación. Como alternativa, puede elegir Diferentes imágenes personalizadas y pegar diferentes URI de imagen de Amazon ECR para cada tipo de trabajador.

CLI

- Cree una aplicación con el parámetro `image-configuration`. EMR sin servidor 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 diferente configuración de imagen para cada tipo de trabajador, utilice el parámetro `worker-type-specifications`.

```
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 parámetro `image-configuration`. EMR sin servidor 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 sin servidor acceda al repositorio de imágenes personalizado

Añada la siguiente política de recursos al repositorio de Amazon ECR para permitir que la entidad principal del servicio EMR sin servidor utilice las solicitudes `get`, `describe` y `download` desde este repositorio.

JSON

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

Como práctica recomendada de seguridad, agregue una clave de condición `aws:SourceArn` a la política del repositorio. La clave de condición global de IAM `aws:SourceArn` garantiza que EMR sin servidor utilice el repositorio solo para un ARN de aplicación. Para obtener más información sobre las políticas de repositorios de Amazon ECR, consulte Creación de 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 versión (por ejemplo, `emr-6.9.0`) de su aplicación.
- EMR sin servidor ignora las instrucciones `[ENTRYPOINT]` o `[CMD]` en el archivo Docker. Utilice las instrucciones comunes en el archivo Docker, como, `[COPY]`, `[RUN]` y `[WORKDIR]`.
- No modifique las variables de entorno `JAVA_HOME`, `SPARK_HOME`, `HIVE_HOME`, `TEZ_HOME` cuando cree una imagen personalizada.

- Las imágenes personalizadas no pueden superar los 10 GB de tamaño.
- Si modifica archivos binarios o jars en las imágenes base de Amazon EMR, es posible que se produzcan errores al iniciar la aplicación o el trabajo.
- El repositorio de Amazon ECR debe estar en la misma Región de AWS que utiliza para lanzar las aplicaciones EMR sin servidor.

Configuración del acceso a la VPC para que las aplicaciones EMR sin servidor se conecten a los datos

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

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

Creación de una aplicación

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

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga los almacenes de datos. La página Crear aplicación enumera todas las VPC que haya elegido Región de AWS.

Subredes

Seleccione las subredes dentro de la VPC que contenga el almacén de datos. La página Crear aplicación enumera todas las subredes de los almacenes de datos de la VPC. Se admiten tanto

subredes públicas como privadas. Puede transferir subredes públicas o privadas a sus aplicaciones. Hay algunas consideraciones asociadas que debe tener en cuenta a la hora de elegir entre una subred pública y una privada.

Para subredes privadas:

- Las tablas de enrutamiento asociadas no deben tener puertas de enlace de Internet.
- De ser necesario, configure las rutas salientes con una puerta de enlace NAT para la conectividad saliente a internet. Para configurar una puerta de enlace NAT, consulte Gateways NAT.
- Para la conectividad de Amazon S3, configure una puerta de enlace NAT o un punto de conexión de VPC. Para configurar un punto de conexión de VPC de S3, consulte Crear un punto de conexión de puerta de enlace.
- Si configura un punto de conexión de VPC de S3 y adjunta una política de punto de conexión para controlar el acceso, siga las instrucciones de Logging for EMR Serverless with managed storage para proporcionar los permisos necesarios a fin de que EMR sin servidor almacene y proporcione los registros de las aplicaciones.
- Para conectarse a otros Servicios de AWS externos a la VPC, como Amazon DynamoDB, configure los puntos de conexión de VPC o una puerta de enlace NAT. Para configurar puntos de conexión de VPC para Servicios de AWS, consulte Work with VPC endpoints.

> **Note**
>
> Cuando se configura una aplicación de Amazon EMR sin servidor en una subred privada, se recomienda configurar también puntos de conexión de VPC para Amazon S3. Si su aplicación de EMR sin servidor se encuentra en una subred privada sin puntos de conexión de VPC para Amazon S3, incurrirá en cargos adicionales de la puerta de enlace de NAT asociados con el tráfico de S3. Esto se debe a que el tráfico entre la aplicación de EMR y Amazon S3 no permanece dentro de su VPC si los puntos de conexión de VPC no están configurados.

Para subredes públicas:

- Tienen una ruta a una puerta de enlace de Internet.
- Debe asegurarse de establecer la configuración adecuada del grupo de seguridad para controlar el tráfico saliente.

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

Cuando usaAWS Config, EMR sin servidor 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` en AWS Config.

> **Note**
>
> Sugerimos seleccionar varias subredes entre varias zonas de disponibilidad. Esto se debe a que las subredes que elija determinan las zonas de disponibilidad disponibles para que se lance una aplicación EMR sin servidor. Cada trabajador consume 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”.

Consideraciones y limitaciones en relación con las subredes

- EMR sin servidor con subredes públicas no es compatible con AWS Lake Formation.
- El tráfico entrante no es compatible con las subredes públicas.

Grupos de seguridad

Elija uno o varios grupos de seguridad que puedan comunicarse con sus almacenes de datos. La página Crear aplicación enumera todos los grupos de seguridad de la VPC. EMR sin servidor asocia estos grupos de seguridad con interfaces de red elástica que se asocian a sus subredes de VPC.

> **Note**
>
> Sugerimos crear un grupo de seguridad independiente para las aplicaciones EMR sin servidor. EMR sin servidor no permite Crear, Actualizar ni Iniciar una aplicación si los grupos de seguridad tienen puertos abiertos a la internet pública en 0.0.0.0/0 o en el rango de ::/0. Esto mejora la seguridad y el aislamiento, y hace que la administración de las reglas de red sea más eficiente. Por ejemplo, bloquea el tráfico inesperado que llega a los trabajadores con direcciones IP públicas. Para comunicarse con los clústeres de Amazon Redshift,

por ejemplo, defina las reglas de tráfico entre los grupos de seguridad Redshift y EMR sin servidor, como se muestra en el ejemplo de la siguiente sección.

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 los grupos de seguridad de EMR sin servidor.

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

2. Agregue una regla para el tráfico saliente desde uno de los grupos de seguridad de 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 los TCP	TCP	5439	`redshift-security-group`

Configurar aplicación

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

Acceda a los detalles de la ejecución del trabajo

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

Prácticas recomendadas para la planificación de subredes

Los recursos de AWS se crean en una subred que es un subconjunto de direcciones IP disponibles en una 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, con cada una de las cuales usando una máscara /17 y 32 768 direcciones IP disponibles. Una subred reside en una zona de disponibilidad y no puede abarcar otras 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 requiere 4 trabajadores de vCPU y se puede escalar verticalmente hasta 4.000 vCPU, su aplicación necesita como máximo 1000 trabajadores para un total de 1000 interfaces de red. Sugerimos crear varias subredes entre varias zonas de disponibilidad. Esto permite a EMR sin servidor volver a intentar su trabajo o aprovisionar la capacidad preinicializada en una zona de disponibilidad diferente en el improbable caso de producirse 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. Ninguna máscara superior a 22 cumple con 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 de /22 en una VPC de máscara de red /16 que se puede asignar a diferentes zonas de disponibilidad. Hay una diferencia de cinco entre las direcciones IP disponibles y las 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	10.0.0.0-10.0.3.255	1 024	1.019
2	10.0.4.0	255.255.252.0/22	10.0.4.0-10.0.7.255	1 024	1.019
3	10.0.8.0	255.255.252.0/22	10.0.8.0-10.0.11.255	1 024	1.019
4	10.0.12.0	255.255.252.0/22	10.0.12.0-10.0.15.255	1 024	1.019

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 trabajadores de vCPU con un límite de escalado de aplicaciones de 4000 vCPU requiere, 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 escalado colectivos de todas sus aplicaciones. Por ejemplo, si tiene 3 aplicaciones que solicitan 4 trabajadores de vCPU y cada una puede escalarse verticalmente hasta 4000 vCPU con una cuota basada en el servicio a nivel de cuenta de 12 000 vCPU, cada subred necesita 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. Puede hacerlo asociando bloques adicionales de enrutamiento entre dominios sin clase (CIDR) con su VPC. Para obtener más información, consulte Asociar bloques de CIDR IPv4 adicionales a su VPC en la Guía del usuario de Amazon VPC.

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

Opciones de la arquitectura de Amazon EMR sin servidor

La arquitectura del conjunto de instrucciones de la aplicación de Amazon EMR sin servidor 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. EMR sin servidor 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 un esfuerzo adicional por su parte.

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 EMR sin servidor. 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 ARM de 64 bits, cámbiese 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)

Los procesadores Graviton de AWS están diseñados a medida por AWS con núcleos ARM Neoverse de 64 bits y aprovechan la arquitectura arm64 (también conocida como Arch64 o ARM de 64 bits). La línea de procesadores Graviton de AWS disponible en EMR sin servidor 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. EMR sin servidor utiliza automáticamente la última generación de procesadores cuando está disponible y no tendrá que implementar nada para actualizarse 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 lanzar una aplicación con procesadores Graviton desde AWS CLI, especifíque `ARM64` como el parámetro `architecture` en la API de `create-application`. 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.

Configuración de las aplicaciones existentes para usar Graviton

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

Para convertir una aplicación existente de x86 a arm64

1. Confirme que está utilizando la última versión principal del AWS CLI/SDK que admite el parámetro `architecture`.
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 a fin de que utilice Graviton, especifique `ARM64` para el parámetro `architectureen` la API de `update-application`.

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

4. Para comprobar que la arquitectura de la CPU de la aplicación es ahora ARM64, utilice la API de `get-application`.

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

5. Cuando haya terminado, reinicie la aplicación.

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

Consideraciones sobre el uso de Graviton

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

Compatibilidad de bibliotecas

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

Para obtener más información, consulte el repositorio AWS Graviton Getting Started en GitHub. Este repositorio contiene recursos esenciales que pueden ayudarte a empezar con Graviton basado en ARM.

Simultaneidad de trabajos y colas para una aplicación EMR sin servidor

A partir de Amazon EMR 7.0.0, se debe especificar el tiempo de espera de la cola de ejecución del trabajo y la configuración de simultaneidad de su aplicación. Al especificar esta configuración, Amazon EMR sin servidor comienza por poner en cola el trabajo y comienza la ejecución en función del uso simultáneo de la aplicación. Por ejemplo, si la simultaneidad de sus trabajos es de 10, solo se ejecutan diez trabajos a la vez en su aplicación. Los trabajos restantes se ponen en cola hasta que finalice uno de los trabajos en ejecución. Si el tiempo de espera de la cola se agota antes, se agota el tiempo de espera de su trabajo. Para obtener más información, consulte Job run states.

Beneficios clave de la simultaneidad y las colas

La simultaneidad de trabajos y la creación de colas ofrecen las siguientes ventajas cuando se requieren muchos envíos de trabajo:

- Ayuda a controlar la ejecución simultánea de los trabajos para utilizar de forma eficiente los límites de capacidad a nivel de aplicación.
- La cola puede contener una ráfaga repentina de envíos de trabajos, con un ajuste de tiempo de espera configurable.

Introducción a la simultaneidad y las colas

Los siguientes procedimientos muestran algunas formas diferentes de implementar la simultaneidad y las colas.

Uso de AWS CLI

1. Cree una aplicación de Amazon EMR sin servidor con tiempo de espera de cola y ejecuciones de trabajos simultáneas:

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

2. Actualice una aplicación para cambiar el tiempo de espera y la simultaneidad de la cola de trabajos:

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

> **Note**
>
> Puede actualizar su aplicación actual para habilitar la simultaneidad de trabajos y la creación de colas. Para ello, la aplicación debe tener una etiqueta de versión emr-7.0.0 o posterior.

Uso de Consola de administración de AWS

En los pasos siguientes, se muestra cómo empezar a utilizar la simultaneidad de trabajos y la creación de colas mediante la Consola de administración de AWS:

1. Vaya a EMR Studio y elija crear una aplicación con la etiqueta de versión EMR-7.0.0 o posterior.
2. En Opciones de configuración de la aplicación, seleccione la opción Usar configuración personalizada.
3. En Configuraciones adicionales hay una sección para la Configuración de la ejecución de trabajos. Seleccione la opción Habilitar la simultaneidad de trabajos para activar la función.
4. Luego, seleccione Ejecuciones de trabajos simultáneos y Tiempo de espera de la cola para configurar la cantidad de ejecuciones de trabajos simultáneos y el tiempo de espera de la cola, respectivamente. Si no introduce valores para estos ajustes, se utilizarán los valores predeterminados.
5. Seleccione Crear aplicación y la aplicación se creará con esta función habilitada. Para verificarlo, vaya al panel de control, seleccione su aplicación y compruebe si la función está habilitada en la pestaña de propiedades.

Tras la configuración, envíe trabajos con esta característica habilitada.

Consideraciones sobre la simultaneidad y las colas

Tenga en cuenta lo siguiente cuando implemente la simultaneidad y las colas:

- La simultaneidad de los trabajos y la cola son compatibles con Amazon EMR 7.0.0 y versiones posteriores.
- La simultaneidad de los trabajos y la cola están habilitadas de forma predeterminada en Amazon EMR 7.3.0 y versiones posteriores.
- No es posible actualizar la simultaneidad de una aplicación en el estado INICIADO.
- El intervalo válido para `maxConcurrentRuns` es de 1 a 1000 y para `queueTimeoutMinutes` de 15 a 720.
- Puede haber un máximo de 2000 trabajos en el estado EN COLA para una cuenta.
- La simultaneidad y las colas se aplican a los trabajos por lotes y en streaming. No se puede usar para trabajos interactivos. Para obtener más información, consulte Run interactive workloads with EMR Serverless through EMR Studio.

Introduzca datos en S3 Express One Zone con EMR sin servidor

Con Amazon EMR 7.2.0 y versiones posteriores, puede utilizar Amazon EMR sin servidor con la clase de almacenamiento de Amazon S3 Express One Zone para mejorar el rendimiento al ejecutar trabajos y cargas de trabajo. S3 Express One Zone es una clase de almacenamiento de Amazon S3 en zona única de alto rendimiento que ofrece acceso constante a los datos en milisegundos de un solo dígito para los datos a los que accede para las aplicaciones más 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 a una acción como `GET`, `LIST` o `PUT` en un objeto de S3, la clase de almacenamiento llama a `CreateSession` en su nombre. Su política de IAM debe permitir el permiso `s3express:CreateSession` para que el conector S3A pueda invocar la API `CreateSession`. Para ver un ejemplo de política con ese permiso, consulte Introducción a S3 Express One Zone.
- Conector S3A: para configurar Spark de modo que pueda acceder a los datos de un bucket de Amazon S3 que utilice la clase de almacenamiento S3 Express One Zone, utilice el conector S3A de Apache Hadoop. Para usar el conector, asegúrese de que todos los URI de S3 usen el esquema `s3a`. De no ser así, cambie 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 punto de conexión de VPC. Agregue el punto de conexión `com.amazonaws.us-west-2.s3express` al punto de conexión de la VPC.
2. Siga Introducción a Amazon EMR sin servidor para crear una aplicación con la etiqueta de versión 7.2.0 o superior de Amazon EMR.
3. Configure su aplicación para que utilice el punto de conexión de VPC recién creado, un grupo de subredes privadas y un grupo de seguridad.
4. Añada el permiso `CreateSession` a su rol de ejecución de trabajos.

   JSON

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

```
  ]
}
```

5. Ejecute su trabajo. Tenga en cuenta que debe usar el esquema `S3A` para acceder a los buckets 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

Una vez que haya preparado su aplicación, envíe trabajos a la dicha aplicación. En esta sección se explica cómo utilizar la AWS CLI para ejecutar estos trabajos. Esta sección también identifica los valores predeterminados para cada tipo de aplicación que está disponible en EMR sin servidor.

Temas

Estados de ejecuciones de trabajos

Al enviar una ejecución de trabajo a una cola de trabajos de Amazon EMR sin servidor, el trabajo pasa al estado `SUBMITTED`. El estado de un trabajo pasa desde el `SUBMITTED` pasando por `RUNNING` hasta que llega a `FAILED`, `SUCCESS` o `CANCELLING`.

Las ejecuciones de trabajos pueden tener los siguientes estados:

Estado	Descripción
Enviado	El estado inicial del trabajo al enviar un trabajo se ejecuta en EMR sin servidor. El trabajo espera a que se programe para la solicitud

Estado	Descripción
	. EMR sin servidor comienza a priorizar y programar la ejecución del trabajo.
En cola	La ejecución de la tarea espera en este estado cuando la ejecución simultánea de tareas a nivel de aplicación esté completamente ocupada. Para obtener más información acerca de la cola y de la simultaneidad, consulte Simultaneidad de trabajos y colas para una aplicación EMR sin servidor.
Pending (Pendiente	El programador evalúa la ejecución del trabajo para priorizar y programar la ejecución de la aplicación.
Programados	EMR sin servidor ha programado la ejecución del trabajo para la aplicación y está asignando recursos para ejecutarlo.
Running	EMR sin servidor 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	EMR sin servidor no pudo enviar la ejecución del trabajo a la aplicación o se completó sin éxito. Consulte `StateDetails` para obtener información adicional sobre este error en el trabajo.
correcto	El trabajo se ha completado correctamente.

Estado	Descripción
Cancelling	La API `CancelJobRun` solicitó la cancelación de la ejecución de la tarea o se agotó el tiempo de espera de la ejecución de la tarea. EMR sin servidor está intentando cancelar el trabajo en la aplicación y liberar los recursos.
Cancelado	La ejecución del trabajo se canceló correctamente y se han liberado los recursos que utilizaba.

Cancelación de la ejecución de un trabajo de EMR sin servidor con período de gracia

En los sistemas de procesamiento de datos, las terminaciones abruptas pueden provocar un desperdicio de recursos, operaciones incompletas y posibles inconsistencias en los datos. Amazon EMR sin servidor le permite especificar un período de gracia al cancelar la ejecución de trabajos. Esta característica permite disponer de tiempo para limpiar adecuadamente y completar la tarea en curso antes de terminar el trabajo.

Al cancelar la ejecución de un trabajo, especifique un período de gracia (en segundos) mediante el parámetro `shutdownGracePeriodInSeconds` durante el cual el trabajo puede realizar operaciones de limpieza antes de su finalización definitiva. El comportamiento y los ajustes predeterminados varían entre los trabajos por lotes y en streaming.

Período de gracia para trabajos por lotes

Para los trabajos por lotes, EMR sin servidor le permite implementar operaciones de limpieza personalizadas que se ejecutan durante el período de gracia. Puede registrar estas operaciones de limpieza como parte del enlace de cierre de la JVM en el código de su aplicación.

Comportamiento predeterminado

El comportamiento predeterminado para el cierre es no tener ningún período de gracia. Consta de las dos siguientes acciones:

- Terminación inmediata

- Los recursos se liberan inmediatamente

Opciones de configuración

Puede especificar una configuración que dé como resultado un cierre estable:

- Intervalo válido para el período de gracia de cierre: de 15 a 1800 segundos (opcional)
- Terminación inmediata (sin período de gracia): 0 segundos

Habilite un cierre estable

Para implementar un cierre estable en los trabajos por lotes, siga estos pasos:

1. Agregue un enlace de apagado en el código de la aplicación que contenga una lógica de cierre personalizada.

   Example in Scala

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

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

   Uso de ShutdownHookManager

   Example in PySpark

   ```
   import atexit

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

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

2. Especifique un período de gracia al cancelar el trabajo para dar tiempo a que se ejecuten los enlaces añadidos anteriormente

Ejemplo

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

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

Período de gracia para los trabajos de streaming

En Spark Structured Streaming, donde los cálculos implican leer o escribir en orígenes de datos externos, las interrupciones abruptas pueden provocar resultados no deseados. Los trabajos de streaming procesan los datos en microlotes y, si se interrumpen estas operaciones a mitad de camino, se puede duplicar el procesamiento en los siguientes intentos. Esto ocurre cuando no se ha escrito el último punto de control del microlote anterior, lo que provoca que los mismos datos se procesen de nuevo cuando se reinicie el trabajo de streaming. Este procesamiento duplicado no solo desperdicia recursos informáticos, sino que también puede afectar a las operaciones comerciales, por lo que es crucial evitar interrupciones abruptas.

EMR sin servidor proporciona soporte integrado para un cierre estable a través de un oyente de consultas de streaming. Esto garantiza la correcta finalización de los microlotes en curso antes de la finalización del trabajo. El servicio gestiona automáticamente el cierre estable entre los microlotes para las aplicaciones de streaming, lo que garantiza que el microlote actual complete el procesamiento, que los puntos de control se escriban correctamente y que el contexto de transmisión finalice de forma limpia sin ingerir nuevos datos durante el proceso de cierre.

Comportamiento predeterminado

- El período de gracia de 120 segundos está activado de forma predeterminada.
- El oyente de consultas de streaming integrado gestiona el cierre estable.

Opciones de configuración

- Intervalo válido para el período de gracia de cierre: de 15 a 1800 segundos (opcional)
- Terminación inmediata: 0 segundos

Habilite el cierre estable

Para implementar un cierre estable de los trabajos de streaming:

Especifique un período de gracia al cancelar el trabajo para dar tiempo a que se complete el microlote en curso.

Ejemplo

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

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

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

Añada enlaces de cierre personalizados (opcional)

Si bien EMR sin servidor gestiona un cierre estable de forma predeterminada a través de su oyente de consultas de streaming integrado, puede implementar opcionalmente una lógica de cierre personalizada para las consultas de streaming individuales. EMR sin servidor registra su elegante oyente de cierre estable con prioridad 60 (mediante ShutdownHookManager). Como los enlaces de mayor prioridad se ejecutan primero, puede registrar sus operaciones de limpieza personalizadas con una prioridad superior a 60 para asegurarse de que se ejecuten antes de que comience el proceso de cierre de EMR sin servidor.

Para añadir un enlace personalizado, consulte el primer ejemplo de este tema, que muestra cómo añadir un enlace de cierre en el código de la aplicación. En este caso, 100 es la prioridad, que es mayor que 60. Por lo tanto, dicho enlace de cierre se ejecuta primero.

> **Note**
>
> Los enlaces de cierre personalizados son opcionales y no son necesarios para una funcionalidad de cierre estable, que EMR sin servidor gestiona automáticamente.

Cargos del período de gracia y duración del lote

Si se utiliza el valor predeterminado para el período de gracia (120 segundos):

- Si la duración del lote es inferior a 120 segundos, solo se le cobrará el tiempo real necesario para completar el lote.
- Si la duración del lote supera los 120 segundos, se le cobrará el período de gracia máximo (120 segundos), pero es posible que la consulta no se cierre correctamente, ya que finalizará de manera forzosa.

Para optimizar los costos y garantizar un cierre estable:

- Para lotes con una duración superior a 120 segundos: considere aumentar el período de gracia para que coincida con la duración del lote
- Para lotes con una duración inferior a 120 segundos: no es necesario ajustar el período de gracia, ya que solo se le cobrará por el tiempo de procesamiento real

Consideraciones

Comportamiento del período de gracia

- El período de gracia proporciona tiempo para que se completen los enlaces de cierre registrados.
- El trabajo finaliza tan pronto como finaliza el enlace de cierre, incluso si es mucho antes del período de gracia.
- Si las operaciones de limpieza superan el período de gracia, el trabajo finalizará forzosamente.

Comportamiento de servicio

- El cierre por período de gracia solo está disponible para los trabajos en estado EN EJECUCIÓN.
- Las solicitudes de cancelación posteriores durante el estado CANCELANDO se ignoran.
- Si EMR sin servidor no puede iniciar el cierre por período de gracia debido a errores internos del servicio:
  - El servicio se volverá a intentar durante un máximo de 2 minutos.
  - Si los reintentos no se realizan correctamente, el trabajo finalizará de manera forzosa.

Facturación

Los trabajos se facturan por los recursos informáticos utilizados hasta que el trabajo se cierre por completo, lo que incluye el tiempo transcurrido durante el período de gracia.

Ejecución de trabajos desde la consola de EMR Studio

Puede enviar ejecuciones de trabajos a aplicaciones EMR sin servidor y acceder a los trabajos desde la consola de EMR Studio. Para crear –o ir hasta– la aplicación EMR sin servidor en EMR Studio, siga las instrucciones de Introducción a la consola.

Enviar un trabajo

En la página Enviar trabajo, envíe 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 Rol del tiempo de ejecución, introduzca el nombre del rol de IAM que la aplicación EMR sin servidor puede asumir para la ejecución del trabajo. Para obtener más información acerca de los roles de tiempos de ejecución, consulte Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.
3. En el campo Ubicación del script, introduzca la ubicación de Amazon S3 del script o el 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 archivo JAR, introduzca el nombre de la clase que es el punto de entrada del trabajo en el campo Clase principal.

5. (Opcional) Introduzca los valores en el resto de los campos.

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

     > **Note**
     >
     > Si especifica los tamaños del controlador y del ejecutor de Spark, tenga 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 del ejecutor y la sobrecarga de memoria juntas no pueden superar la memoria de trabajo. Por ejemplo, la `spark.executor.memory` máxima en un trabajador de 30 GB debe ser de 27 GB.

   - Configuración del trabajo: especifique cualquier configuración de trabajo en este campo. Puede usar estas configuraciones del trabajo para anular las configuraciones predeterminadas de las aplicaciones.
   - Ajustes adicionales: active o desactive el catálogo de datos de Glue de AWS como metaalmacén y modifique los ajustes del registro de la aplicación. Para obtener más información sobre las configuraciones del metaalmacén, consulte Configuración de metaalmacenes para EMR sin servidor. Para obtener más información sobre las opciones de registro de aplicaciones, consulte Almacenamiento 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 Rol del tiempo de ejecución, introduzca el nombre del rol de IAM que la aplicación EMR sin servidor puede asumir para la ejecución del trabajo.

3. En el campo Ubicación del script, introduzca la ubicación de Amazon S3 del script o el JAR que desee ejecutar. Para los trabajos de Hive, el script debe ser un archivo Hive (`.sql`).
4. (Opcional) Introduzca los valores en el resto de los campos.
   - 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 Spark: expanda la sección de propiedades de Hive e introduzca cualquier parámetro de configuración de Hive en ese campo.
   - Configuración del trabajo: especifique cualquier configuración del trabajo. Puede usar estas configuraciones del 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 configuración del `hive-site`.

     ```
     {
         "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 catálogo de datos de Glue de AWS como metaalmacén y modifique los ajustes del registro de la aplicación. Para obtener más información sobre las configuraciones del metaalmacén, consulte Configuración de metaalmacenes para EMR sin servidor. Para obtener más información sobre las opciones de registro de aplicaciones, consulte Almacenamiento de registros.
   - Etiquetas: asigne cualquier etiqueta personalizada a la aplicación.
5. Seleccione Enviar el trabajo.

Acceso a las ejecuciones de trabajos

En la pestaña Ejecuciones de trabajo de la página Detalles de una aplicación, puede acceder a las ejecuciones de trabajos y realizar las siguientes acciones para las ejecuciones de trabajos.

Cancelar trabajo: para cancelar una ejecución de trabajo que esté en el estado `RUNNING`, elija esta opción. Para obtener más información sobre las transiciones de ejecución de trabajos, consulte Estados de ejecuciones de trabajos.

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

Ejecutar trabajos desde AWS CLI

En la AWS CLI, puede crear, describir y eliminar trabajos individuales. También puede enumerar todos sus trabajos para acceder a ellos 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 Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor. Para ver ejemplos de Hive, consulte Uso de configuraciones de Hive al ejecutar trabajos de EMR sin servidor. Este comando devuelve su `application-id`, ARN y un `job-id` nuevo.

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

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

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

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 acceder a todos sus trabajos, especifique el número máximo de trabajos a los que desea acceder, hasta 50. En el ejemplo siguiente, se especifica que desea acceder a 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 AWS CLI, consulte la referencia de la API de EMR sin servidor.

Política de IAM de ejecución

Puede especificar una política de IAM de ejecución, además de un rol de ejecución, al enviar las ejecuciones de trabajos en EMR sin servidor. Los permisos resultantes que asume la ejecución del trabajo son la intersección de los permisos del rol de ejecución y la política de IAM de ejecución especificada.

Introducción

Pasos para usar la política de IAM de ejecución:

Cree una aplicación `emr-serverless` o use una existente y, a continuación, ejecute la siguiente CLI de <shared id="AWS"/> para iniciar una ejecución de trabajo con una política de IAM integrada:

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

Ejemplos de comandos de la CLI

Si tenemos la siguiente política almacenada en el archivo `policy.json` de la máquina:

JSON

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

```
  ]
}
```

Podemos iniciar un trabajo con esta política mediante el siguiente comando de AWS CLI:

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

También puede usar las políticas de AWS y las administradas por el cliente, especificándolas mediante sus ARN:

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

También es posible especificar los ARN de una política de IAM integrada y de una política administrada en la misma solicitud:

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

Notas importantes

- El campo `policy` de la `execution-role-policy` puede tener una longitud máxima de 2048 caracteres.
- La cadena de política de IAM integrada que se especifica en el campo `policy` de la `execution-iam-policy` debe ajustarse al estándar de cadenas json y no debe omitir líneas nuevas ni comillas como en el ejemplo anterior.
- Se puede especificar una lista de hasta 10 ARN de políticas administradas como valor para el campo `policyArns` de la `execution-iam-policy`.
- Los ARN de políticas administradas deben ser una lista de AWS válida o un ARN de política administrada por el cliente válidos. Cuando se especifica un ARN de política administrada por el cliente, la política debe pertenecer a la misma cuenta de AWS que la JobRun de EMR-S.
- Cuando se usan una política de IAM integrada y políticas administradas, el texto simple que se utilice para ambas políticas combinadas no puede superar los 2048 caracteres.
- Los permisos resultantes que asume JobRun son la intersección de los permisos del rol de ejecución y la política de IAM de ejecución especificada.

Intersección de políticas

Los permisos resultantes que asume la ejecución del trabajo son la intersección de los permisos del rol de ejecución y la política de IAM de ejecución especificada. Lo que significa que cualquier permiso requerido tendrá que especificarse en ambos lugares para que JobRun funcione. Sin embargo, es posible especificar una declaración de autorización general adicional en la política integrada para cualquier permiso que no se pretenda actualizar o sobrescribir.

Ejemplo

Dada la siguiente política de rol de IAM de ejecución:

JSON

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

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

Y la siguiente política de IAM insertada:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

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

Los permisos resultantes que asume JobRun son los siguientes:

JSON

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

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

Uso de discos optimizados para reproducción aleatoria

Con las versiones 7.1.0 y posteriores de Amazon EMR, use discos optimizados para la reproducción aleatoria cuando ejecute trabajos de Apache Spark o Hive a fin de 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 mayores IOPS (operaciones de E/S por segundo) para acelerar el movimiento de los datos y reducir la latencia durante las operaciones de reproducción aleatoria. Los discos optimizados para la reproducción aleatoria le permiten conectar discos de hasta 2 TB por trabajador, por lo que puede configurar la capacidad adecuada para sus requisitos de cargas de trabajo.

Ventajas principales

Los discos optimizados para la reproducción aleatoria proporcionan las siguientes ventajas.

- Alto rendimiento de IOPS: los discos optimizados para la reproducción aleatoria proporcionan más IOPS que los discos estándar, lo que permite una transferencia de datos más eficiente y rápida durante las tareas de Spark y Hive y otras cargas de trabajo de uso intensivo en su reproducción.

- Tamaño de disco más grande: los discos optimizados para la reproducción aleatoria 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 EMR sin servidor versión 7.1.0 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. Configure su trabajo de Spark para que incluya los parámetros `spark.emr-serverless.driver.disk.type` o `spark.emr-serverless.executor.disk.type` a fin de que se ejecute con discos optimizados para la reproducción aleatoria. Puede utilizar uno o ambos parámetros en función de su 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, consulte Spark job properties.

Hive

1. Cree una aplicación EMR sin servidor versión 7.1.0 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 `hive.tez.disk.type` a fin de que se ejecute con discos optimizados para la reproducción aleatoria. Puede utilizar uno o ambos parámetros en función de 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, consulte Propiedades de trabajos de Hive.

Configuración de una aplicación con capacidad preinicializada

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

- 5 controladores de 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, escala verticalmente los trabajadores bajo demanda hasta la capacidad máxima de 400 vCPU y 1024 GB de memoria. Si lo desea, puede omitir la capacidad del trabajador del `DRIVER` o del `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 para procesar datos transmitidos de forma continua

Un trabajo de streaming en EMR sin servidor es un modo de trabajo que le permite analizar y procesar los datos de streaming casi en tiempo real. Estos trabajos de larga duración recopilan los datos de streaming y procesan continuamente los resultados 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 casi en tiempo real, la detección de fraudes y los motores de recomendaciones. Los trabajos de streaming de EMR sin servidor proporcionan optimizaciones, como la resiliencia de los trabajos integrada, la monitorización en tiempo real, la administración mejorada de registros y la integración con conectores de streaming.

Estos son algunos casos de uso con trabajos de streaming:

- Análisis casi en tiempo real: los trabajos de streaming en Amazon EMR sin servidor le permiten procesar los datos de streaming casi en tiempo real, de modo que puede realizar análisis en tiempo real de flujos de datos continuos, como datos de registros, datos de sensores o flujos de clics para obtener información y tomar decisiones oportunas en función de la información más reciente.
- Detección de fraudes: use los trabajos de streaming para detectar fraudes casi en tiempo real en transacciones financieras, operaciones con tarjetas de crédito o actividades en línea al analizar los flujos de datos e identificar anomalías y patrones sospechosos a medida que se producen.
- Motores de recomendación: 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 redes sociales en tiempo real, como tuits, publicaciones o comentarios, lo que permite a las organizaciones supervisar las tendencias, analizar las opiniones y gestionar la reputación de la marca en casi tiempo real.
- Análisis del Internet de las cosas (IoT): los trabajos de streaming pueden permitir gestionar y analizar los flujos de datos de alta velocidad de maquinaria conectada y dispositivos y sensores del IoT, de modo que usted pueda ejecutar la detección de anomalías, el mantenimiento predictivo y otros casos de uso de análisis del IoT.
- Análisis del flujo de clics: los trabajos de streaming pueden 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 pueden también procesarlos datos del registro de servidores, aplicaciones o dispositivos de red. Esto le permite detectar anomalías, solucionar problemas y obtener información acerca del estado y el rendimiento del sistema.

Ventajas principales

Los trabajos de streaming en EMR sin servidor proporcionan automáticamente resiliencia al trabajo, siendo una combinación de los siguientes factores:

- Reintento automático: EMR sin servidor reintenta automáticamente cualquier trabajo que haya fallado sin ninguna intervención manual por su parte.
- Resiliencia de la zona de disponibilidad (AZ): EMR sin servidor cambia automáticamente los trabajos de streaming a una AZ en buen estado si la AZ original tiene problemas.
- Administración de registros:
  - Rotación de registros: para una administración más eficiente del almacenamiento en disco, EMR sin servidor rota periódicamente los registros para trabajos de streaming 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 administrada. La compactación también mejora la experiencia de depuración cuando se utiliza el servidor de historial de Spark administrado.

Fuentes de datos y receptores de datos compatibles

EMR sin servidor funciona con varias fuentes de datos de entrada y receptores de datos de salida:

- Fuentes de datos de entrada admitidas: Amazon Kinesis Data Streams, Amazon Managed Streaming para Apache Kafka y los clústeres de Apache Kafka autogestionado. De forma predeterminada, las versiones 7.1.0 y posteriores de Amazon EMR incluyen el conector de Amazon Kinesis Data Streams, por lo que no es necesario crear ni descargar ningún paquete adicional.
- Receptores de datos de salida compatibles: tablas de del catálogo de datos de Glue de AWS, Amazon S3, Amazon Redshift, MySQL, PostgreSQL Oracle, Oracle, Microsoft SQL, Apache Iceberg, Delta Lake y Apache Hudi.

Consideraciones y limitaciones

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

- Los trabajos de streaming son compatibles con las versiones 7.1.0 y posteriores de Amazon EMR.
- EMR sin servidor espera que los trabajos de streaming se ejecuten durante mucho tiempo, por lo que no 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.
- EMR sin servidor reintenta los trabajos de streaming de forma indefinida y usted no se puede 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 umbral predeterminado es de cinco intentos fallidos en una hora. Puede configurar este umbral para que esté entre 1 y 10 intentos. Para obtener más información, consulte Resiliencia de trabajos.
- Los trabajos de streaming tienen puntos de comprobación para guardar el estado y el progreso del tiempo de ejecución, por lo que EMR sin servidor puede reanudar el trabajo de streaming desde el último punto de comprobación. Para obtener más información, consulte Recovering from failures with Checkpointing en la documentación de Apache Spark.

Introducción a trabajos de streaming

Consulte las siguientes instrucciones para obtener información sobre cómo empezar a realizar trabajos de streaming.

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

```
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 admitidos

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

Los siguientes son los conectores de streaming admitidos:

Conector Amazon Kinesis Data Streams

El conector Amazon Kinesis Data Streams para Apache Spark permite crear aplicaciones y canalizaciones de streaming que consumen datos del Amazon Kinesis Data Streams, además de escribir en él. El conector permite aumentar el consumo del ventilador con una velocidad de lectura específica de hasta 2 MB/segundo por partición. De forma predeterminada, Amazon EMR sin servidor 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 página spark-sql-kinesis-connector 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, configure la aplicación EMR sin servidor con acceso a la VPC y usar un punto de conexión de VPC para permitir el acceso privado, o usar una puerta de enlace NAT para obtener el acceso público. Para obtener más información, consulte Configuración del acceso a la VPC. Asimismo, debe asegurarse de que su rol de tiempo de ejecución de trabajos tenga los permisos de lectura y escritura necesarios para obtener acceso a los flujos de datos requeridos. Para obtener más información sobre cómo configurar un rol de tiempo de ejecución de tareas, consulte Job runtime roles for Amazon EMR Serverless. Para obtener una lista completa de todos los permisos necesarios, consulte la página spark-sql-kinesis-connector en GitHub.

Conector Kafka de Apache

El conector Apache Kafka para el streaming estructurado 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 autogestionado y Amazon Managed Streaming para Apache Kafka. Para obtener más información sobre el conector, consulte Structured Streaming + Kafka Integration Guide en la documentación de Apache Spark.

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

```
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 sin servidor y de la versión de Spark correspondiente. Para encontrar la versión correcta de Kafka, consulte Structured Streaming + Kafka Integration Guide.

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

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

Para utilizar el conector Kafka y la biblioteca de autenticación de IAM de Amazon MSK, configure la aplicación EMR sin servidor con acceso a la VPC. Sus subredes deben tener acceso a Internet y utilizar una puerta de enlace NAT para acceder a las dependencias de Maven. Para obtener más información, consulte Configuración del acceso a la 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 para Apache Kafka.

Administración de registros de trabajos en streaming

Los trabajos de streaming admiten la rotación de registros para los registros de aplicaciones y eventos de Spark, así como la compactación de registros para los registros de eventos de Spark. Esto le ayuda a administrar sus recursos con eficacia.

Rotación de registros

Los trabajos de streaming admiten la rotación de registros para los registros de aplicaciones y registros de 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 registros

Los trabajos de streaming también admiten la compactación de registros para los registros de eventos de Spark siempre que haya registros administrados disponibles. Para obtener más información sobre los registros administrados, consulte Logging with managed storage. Los trabajos de streaming pueden durar bastante, 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 IU de la aplicación Spark. Este proceso puede provocar elevados costes y latencias, 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 servidor de historial de Spark no necesita cargar más de 1 GB de registros de eventos en ningún momento. Para obtener más información, consulte Monitoring and Instrumentation en la documentación de Apache Spark.

Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor

Puede ejecutar trabajos de Spark en una aplicación con el parámetro `type` establecido en `SPARK`. Los trabajos deben ser compatibles con la versión de Spark compatible con la versión de lanzamiento de Amazon EMR. Por ejemplo, cuando ejecuta trabajos en una aplicación con la versión 6.6.0 de Amazon EMR, 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 lanzamiento de Amazon EMR sin servidor.

Parámetros de trabajos de Spark

Al utilizar la API `StartJobRun` para ejecutar un trabajo de Spark, especifique los siguientes parámetros.

Parámetros necesarios

- Rol de tiempo de ejecución de trabajos de Spark
- Parámetro del controlador de trabajos de Spark
- Parámetro de anulación de configuración de Spark
- Optimización de asignación de recursos dinámicos de Spark

Rol de tiempo de ejecución de trabajos de Spark

Utilice **`executionRoleArn`** para especificar el ARN del rol de IAM que la aplicación utiliza para ejecutar los trabajos de Spark. Este rol debe tener los siguientes permisos:

- Lectura de los buckets de S3 u otras fuentes de datos en las que residen sus datos
- Lectura de los buckets o prefijos de S3 donde reside tu script PySpark o archivo JAR
- Escritura en los buckets de S3 donde desee escribir el resultado final
- Escritura en los registros en un prefijo o bucket de S3 que especifique `S3MonitoringConfiguration`
- Acceso a las claves de KMS si usa claves de KMS para cifrar los datos en el bucket de S3
- Acceso al catálogo de datos de Glue de AWS si usa SparkSQL

Si su trabajo de Spark lee o escribe datos en –o desde– otras fuentes de datos, especifique los permisos adecuados en este rol de IAM. Si no proporciona estos permisos al rol de IAM, es posible que el trabajo no funcione. Para obtener más información, consulte Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor y Almacenamiento de registros.

Parámetro del controlador de trabajos de Spark

Utilice **`jobDriver`** para proporcionar datos al trabajo. El parámetro del controlador de trabajo solo acepta un valor para el tipo de trabajo que desee ejecutar. Para un trabajo de Spark, el valor del

parámetro es `sparkSubmit`. Puede usar este tipo de trabajo para ejecutar Scala, Java, PySpark, y cualquier otro trabajo compatible mediante Spark-submit. Los trabajos de Spark tienen los parámetros siguientes:

- **`sparkSubmitParameters`**: son los parámetros de Spark adicionales que desea enviar al trabajo. Use este parámetro para anular las propiedades predeterminadas de Spark, como la memoria del controlador o el número de ejecutores, como aquellos definidos en los argumentos `--conf` o `--class`.
- **`entryPointArguments`**: se trata de un conjunto de argumentos que desea pasar a su archivo Python o JAR principal. Debería manejar la lectura de estos parámetros mediante su código de punto de entrada. Separe cada argumento del conjunto con una coma.
- **`entryPoint`**: esta es la referencia en Amazon S3 al archivo Python o JAR principal que desea ejecutar. Si está ejecutando un JAR de Scala o Java, especifique la clase de entrada principal en `SparkSubmitParameters` utilizando el argumento `--class`.

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

Parámetro de anulación de configuración de Spark

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

- **`monitoringConfiguration`**: utilice este campo para especificar la URL de Amazon S3 (`s3MonitoringConfiguration`) en la que quiere que el trabajo del EMR sin servidor almacene los registros de su trabajo de Spark. Asegúrese de crear este bucket con la misma Cuenta de AWS que aloja su aplicación y en la misma 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 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. Es posible especificar varias clasificaciones para varias aplicaciones en un solo objeto JSON.

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

Si usa la misma configuración en una anulación de aplicación y en los parámetros de envío de Spark, los parámetros de envío de Spark tendrán prioridad. Las configuraciones se clasifican según prioridad de la siguiente manera, de mayor a menor:

- Configuración que EMR sin servidor proporciona cuando crea `SparkSession`.
- Configuración que usted proporciona como parte de los `sparkSubmitParameters` con el argumento `--conf`.
- Configuración que usted proporciona como parte de la aplicación que se anula al iniciar un trabajo.
- Configuración que usted proporciona como parte de la `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, consulte Configuración predeterminada de aplicación para EMR sin servidor.

Optimización de asignación de recursos dinámicos de Spark

Utilice `dynamicAllocationOptimization` para optimizar el uso de recursos en EMR sin servidor. Al establecer esta propiedad `true` en la clasificación de configuración de Spark, EMR sin servidor 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 sin servidor crea y libera trabajadores. Al hacerlo, EMR sin servidor reutiliza de manera más óptima a los trabajadores en todas las etapas, lo que reduce los costes al ejecutar trabajos con varias etapas y, al mismo tiempo, mantiene el mismo rendimiento.

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

A continuación se muestra una clasificación de configuración de ejemplo 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 ha activado la asignación dinámica de recursos.
- Para lograr la mejor rentabilidad, le sugerimos configurar un límite de escalado superior para los trabajadores, y utilice la configuración a nivel de trabajo `spark.dynamicAllocation.maxExecutors` o la configuración de capacidad máxima a nivel de aplicación en función de la carga de trabajo.
- Es posible que no vea una mejora en los costes de los trabajos más sencillos. Por ejemplo, si su 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 podrían experimentar una regresión en el tiempo de ejecución del trabajo. Dado que EMR sin servidor utiliza los recursos de manera más eficiente, es posible que haya menos trabajadores disponibles para las etapas más grandes y, por lo tanto, un tiempo de ejecución más prolongado.

Propiedades de trabajo de Spark

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

Propiedades opcionales y valores predeterminados de Spark

Clave	Descripción	Valor predeterminado
`spark.archives`	Lista separada por comas de archivos que Spark extrae en el directorio de trabajo de cada ejecutor. Los tipos de archivos admitidos son `.jar`, `.tar.gz`, `.tgz` y `.zip`. Para especificar el nombre del directorio que se va a extraer, añada # después del nombre del archivo que desee extraer. Por ejemplo, `file.zip#directory` .	`NULL`
`spark.authenticate`	Opción que activa la autenticación de las conexiones internas de Spark.	`TRUE`
`spark.driver.cores`	La cantidad de núcleos que utiliza el controlador.	4
`spark.driver.extraJavaOptions`	Opciones de Java adicionales para el controlador de 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 escala o reduce verticalmente el número de ejecutores registrados en la aplicación, en función de la carga de trabajo.	`TRUE`

Clave	Descripción	Valor predeterminado
`spark.dynamicAllocation.executorIdleTimeout`	El tiempo que un ejecutor puede permanecer inactivo antes de que Spark lo elimine. Esto solo se aplica si activa la asignación dinámica.	60 s
`spark.dynamicAllocation.initialExecutors`	El número inicial de ejecutores que se ejecutarán si activa la asignación dinámica.	3
`spark.dynamicAllocation.maxExecutors`	El límite superior del número de ejecutores si activa la asignación dinámica.	Para 6.10.0 y versiones posteriores, `infinity` Para 6.9.0 y versiones anteriores, `100`
`spark.dynamicAllocation.minExecutors`	El límite superior del número de ejecutores si 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 de Spark.	20 G
`spark.emr-serverless.driverEnv.` *`[KEY]`*	Opción que añade variables de entorno al controlador de Spark.	`NULL`
`spark.emr-serverless.executor.disk`	El disco ejecutor de Spark.	20 G

Clave	Descripción	Valor predeterminado
`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 de Spark.	Estándar
`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 usa cada ejecutor.	4
`spark.executor.extraJavaOptions`	Opciones de Java adicionales para el ejecutor de Spark.	`NULL`
`spark.executor.instances`	El número de contenedores de ejecutores de Spark que se asignarán.	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`	Lista de archivos separados por comas que se colocarán en el directorio de trabajo de cada ejecutor. Puede acceder a las rutas de estos archivos en el ejecutor con `SparkFiles.get(` *`fileName`*`)`.	`NULL`

Clave	Descripción	Valor predeterminado
`spark.hadoop.hive.metastore.client.factory.class`	La clase de implementación del metaalmacén de Hive.	`NULL`
`spark.jars`	Archivos jar adicionales para añadirlos a la ruta de clases de tiempo de ejecución del controlador y los ejecutores.	`NULL`
`spark.network.crypto.enabled`	Opción que activa el cifrado RPC basado en AES. 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 archivos `.zip`, `.egg` o `.py` para colocarlos en la `PYTHONPATH` en las aplicaciones de Python.	`NULL`

En la tabla siguiente se enumeran los parámetros de envío predeterminados de Spark.

Parámetros de envío predeterminados de Spark

Clave	Descripción	Valor predeterminado
`archives`	Lista separada por comas de archivos que Spark extrae en el directorio de trabajo de cada ejecutor.	`NULL`

Clave	Descripción	Valor predeterminado
`class`	La clase principal de la aplicación (para aplicaciones Java y Scala).	`NULL`
`conf`	Una propiedad de configuración arbitraria de Spark.	`NULL`
`driver-cores`	La cantidad 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 usa cada ejecutor.	4
`executor-memory`	La cantidad de memoria que utiliza el ejecutor.	14 G
`files`	Lista de archivos separados por comas que se colocarán 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 separada por comas de archivos jar para incluirlos en las rutas de clases del controlador y el 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 archivos `.zip`, `.egg` o `.py` para colocarlos en la `PYTHONPATH` para las aplicaciones de Python.	`NULL`
`verbose`	Opción que activa una salida de depuración adicional.	`NULL`

Prácticas recomendadas de configuración de recursos

Configuración de los recursos del controlador y del ejecutor mediante la API StartJobRun

> **Note**
>
> Los núcleos y las propiedades de memoria del controlador y ejecutor de Spark, si se especifican, deben indicarse directamente en la solicitud de la API StartJobRun.

La configuración de los recursos de esta manera garantiza que EMR sin servidor pueda asignar los recursos correctos antes de ejecutar el trabajo. Esto contrasta con la configuración que se proporciona en el script de usuario, como en el archivo .py o .jar, que se evalúa demasiado tarde, ya que los controladores y ejecutores a veces se aprovisionan previamente antes de que comience la ejecución del script. Se admiten dos formas de configurar estos recursos durante el envío del trabajo:

Opción 1: use sparkSubmitParameters

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

```
 }
```

Opción 2: use ConfigurationOverrides para la clasificación de spark-defaults

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

Ejemplos de Spark

El siguiente ejemplo muestra cómo utilizar la API `StartJobRun` para ejecutar un script de Python. Para obtener un tutorial integral en el que se utiliza este ejemplo, consulte Introducción a Amazon EMR sin servidor. Puede encontrar ejemplos adicionales de cómo ejecutar trabajos de PySpark y agregar dependencias de Python en el repositorio GitHub de Ejemplos de EMR sin servidor.

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

El siguiente ejemplo muestra cómo utilizar la API `StartJobRun` para ejecutar un JAR de Spark.

```
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"
        }
    }'
```

Uso de configuraciones de Hive al ejecutar trabajos de EMR sin servidor

Puede ejecutar trabajos de Hive en una aplicación con el parámetro `type` establecido en `HIVE`. Los trabajos deben ser compatibles con la versión de Hive compatible con la versión de lanzamiento de Amazon EMR. Por ejemplo, cuando ejecuta trabajos en una aplicación con Amazon EMR versión 6.6.0, 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 lanzamiento de Amazon EMR sin servidor.

Parámetros de los trabajos de Hive

Al utilizar la API `StartJobRun` para ejecutar un trabajo de Hive, especifique los siguientes parámetros.

Parámetros necesarios

- Rol de tiempo de ejecución de los trabajos de Hive
- Parámetros del controlador de los trabajos de Hive
- Parámetro de anulación de la configuración de Hive

Rol de tiempo de ejecución de los trabajos de Hive

Utilice **`executionRoleArn`** para especificar el ARN del rol de IAM que la aplicación utiliza para ejecutar los trabajos de Hive. Este rol debe tener los siguientes permisos:

- Lectura de los buckets de S3 u otras fuentes de datos en las que residen sus datos
- Lectura de los prefijos o buckets de S3 donde residen el archivo de consulta de Hive y el archivo de consulta de init
- Lectura y escritura en los buckets de S3 donde se residen el directorio de Hive Scratch y el directorio de almacén de Hive Metastore
- Escritura en los buckets de S3 donde desee escribir el resultado final
- Escritura en los registros en un bucket o prefijo de S3 que especifique `S3MonitoringConfiguration`
- Acceso a las claves de KMS si usa claves de KMS para cifrar los datos en el bucket de S3
- Acceso al Catálogo de datos Glue de AWS.

Si su trabajo de Hive lee o escribe datos en –o desde– otras fuentes de datos, especifique los permisos adecuados en este rol de IAM. Si no proporciona estos permisos al rol de IAM, es posible que su trabajo no funcione. Para obtener más información, consulta Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.

Parámetros del controlador de los trabajos de Hive

Utilice **`jobDriver`** para proporcionar datos al trabajo. El parámetro del controlador de trabajo solo acepta un valor para el tipo de trabajo que desee ejecutar. Cuando se especifica `hive` como tipo de trabajo, EMR sin servidor pasa una consulta de Hive al parámetro `jobDriver`. Los trabajos de Hive tienen los parámetros siguientes:

- **`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 Hive de init. Hive ejecuta este archivo antes de la consulta y puede usarlo para inicializar tablas.

Parámetro de anulación de la configuración de Hive

Utilice **`configurationOverrides`** para anular las propiedades de configuración a nivel de monitorización y de aplicación. Este parámetro es un objeto JSON con los dos campos siguientes:

- **`monitoringConfiguration`**: utilice este campo para especificar la URL de Amazon S3 (`s3MonitoringConfiguration`) donde desea que el trabajo de EMR sin servidor almacene los registros de su trabajo de Hive. Asegúrese de crear este bucket con la mismo Cuenta de AWS que aloja su aplicación y en la misma Región de AWS donde se ejecuta su trabajo.
- **`applicationConfiguration`**: puede proporcionar un objeto de configuración en este campo para anular las configuraciones para aplicaciones predeterminadas. Puede utilizar una sintaxis abreviada para proporcionar la configuración o 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. Es posible especificar varias clasificaciones para varias aplicaciones en un solo objeto JSON.

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

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

- Configuración que se proporciona como parte de los parámetros de Hive con `--hiveconf` *`property=value`*.
- Configuración que se proporciona como parte de la aplicación que se anula al iniciar un trabajo.
- Configuración que se proporciona como parte de la `runtimeConfiguration` al crear una aplicación.
- Configuraciones optimizadas que Amazon EMR asigna a 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, consulte Configuración predeterminada de aplicación para EMR sin servidor.

Propiedades de los trabajos de Hive

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

Propiedades obligatorias de los trabajos de Hive

Ajuste	Descripción
`hive.exec.scratchdir`	La ubicación de Amazon S3 en la que EMR sin servidor 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 tablas administradas en Hive.

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

Propiedades opcionales y valores predeterminados de Hive

Ajuste	Descripción	Valor predeterminado
`fs.s3.customAWSCredentialsProvider`	El proveedor de credenciales de AWS que desea usar.	com.amazonaws.auth.DefaultAWSCredentialsProviderChain
`fs.s3a.aws.credentials.provider`	El proveedor de credenciales de AWS que desea usar con un sistema de archivos S3A.	com.amazonaws.auth.DefaultAWSCredentialsProviderChain
`hive.auto.convert.join`	Opción que activa la conversión automática de las uniones comunes en mapjoins, en función del tamaño del archivo de entrada.	`TRUE`

Ajuste	Descripción	Valor predeterminado
`hive.auto.convert.join.noconditionaltask`	Opción que activa la optimización cuando Hive convierte una unión común en un mapjoin en función del tamaño del archivo de entrada.	`TRUE`
`hive.auto.convert.join.noconditionaltask.size`	Una unión se convierte directamente en un mapjoin con 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 establece 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 metaalmacén. Para obtener estadísticas básicas, establezca `hive.stats.autogather` en TRUE. Para obtener una colección de consultas más avanzada, ejecute `analyze table queries`.	`TRUE`

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

Ajuste	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 modo estricto, especifique al menos una partición estática en caso de que sobrescriba todas las particiones de forma accidental. 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 particiones dinámicas que Hive crea en cada nodo mapeador y reductor.	100

Ajuste	Descripción	Valor predeterminado
`hive.exec.orc.split.strategy`	Se espera que sea 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 inicial especificada en la propiedad `hive.exec.scratchdir` .	`.hive-staging`

Ajuste	Descripción	Valor predeterminado
hive.fetch.task.conversion	Se espera que sea uno de los siguientes valores: NONE, MINIMAL o MORE. Hive puede convertir consultas seleccionadas en una sola tarea FETCH. Esto minimiza la latencia.	MORE
hive.groupby.position.alias	Opción que hace que Hive utilice un alias de posición de columna en las instrucciones GROUP BY.	FALSE
hive.input.format	El formato de entrada predeterminado. Establézcalo en HiveInputFormat si surgen 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 se admite con Amazon EMR 6.10.x y versiones anteriores.	TRUE

Ajuste	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 computación y de prioridad de tareas.	1 000
`hive.merge.mapfiles`	Opción que hace que los archivos pequeños se fusionen al final de un trabajo de solo mapas.	`TRUE`
`hive.merge.size.per.task`	El tamaño de los archivos fusionados al final del trabajo.	256000000
`hive.merge.tezfiles`	Opción que activa una fusión de archivos pequeños al final de un Tez DAG.	`FALSE`
`hive.metastore.client.factory.class`	El nombre de la clase de fábrica que produce los objetos que implement an la interfaz `IMetaStoreClient` .	`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`
`hive.metastore.glue.catalogid`	Si el catálogo de datos Glue de AWS actúa como un metaalmacén pero se ejecuta en una Cuenta de AWS diferente a la de los trabajos, el ID de la Cuenta de AWS en el que se ejecutan los trabajos.	`NULL`

Ajuste	Descripción	Valor predeterminado
`hive.metastore.uris`	El URI de Thrift que el cliente del metaalmacén utiliza para conectarse al metaalmacén remoto.	`NULL`
`hive.optimize.ppd`	Opción que activa la pulsación de predicados.	`TRUE`
`hive.optimize.ppd.storage`	Opción que activa la pulsación 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 instrucciones `ORDER BY`.	`TRUE`
`hive.prewarm.enabled`	Opción que activa el precalentamiento del contenedor para Tez.	`FALSE`
`hive.prewarm.numcontainers`	El número de contenedores que se deben precalentar para el té.	10
`hive.stats.autogather`	Opción que hace que Hive recopile estadísticas básicas automáticamente durante el comando `INSERT OVERWRITE` .	`TRUE`

Ajuste	Descripción	Valor predeterminado
`hive.stats.fetch.column.stats`	Opción que desactiva la búsqueda de estadísticas de columnas desde el metaalmacén. 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 comandos Analyze `partialscan` y `noscan` para las tablas particionadas. Esto solo se aplica a los formatos de archivo que implementan `StatsProvidingRecordReader` (como ORC).	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 las comprobaciones de seguridad de tipo estricto y desactiva la comparación de `bigint` con tanto `string` como `double`.	`TRUE`

Ajuste	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 característica de paralelismo del autorreductor Tez. Hive sigue calculando los tamaños de los datos y establece cálculos de paralelismo. Tez toma muestras de los tamaños de salida de los vértices de la fuente y ajusta los cálculos en tiempo de ejecución según sea necesario.	`TRUE`
`hive.tez.container.size`	La cantidad de memoria que se utilizará por proceso de tarea de Tez.	6144
`hive.tez.cpu.vcores`	La cantidad de núcleos que se utilizarán 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`

Ajuste	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 en el lado de reducción de ejecución de una consulta.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	El nombre de clase de controlador para un metaalmacén de JDBC.	`org.apache.derby.jdbc.EmbeddedDriver`
`javax.jdo.option.ConnectionPassword`	La contraseña asociada a una base de datos de metaalmacén.	`NULL`
`javax.jdo.option.ConnectionURL`	La cadena para una conexión de JDBC para un metaalmacén de JDBC.	`jdbc:derby:;databaseName=metastore_db;create=true`
`javax.jdo.option.ConnectionUserName`	El nombre de usuario asociado a una base de datos de metaalmacén.	`NULL`

Ajuste	Descripción	Valor predeterminado
`mapreduce.input.fileinputformat.split.maxsize`	El tamaño máximo de una división durante la computación de división cuando el formato de entrada es `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`. Un valor de 0 indica que no hay ningún límite.	0
`tez.am.dag.cleanup.on.completion`	Opción que activa la limpieza de los datos aleatorios cuando finaliza el DAG.	`TRUE`
`tez.am.emr-serverless.launch.env.[KEY]`	Opción para establecer la variable de entorno *KEY* en el proceso de Tez AM. Para Tez AM, este valor anula el valor `hive.emr-serverless.launch.env.[KEY]`.	
`tez.am.log.level`	El nivel de registro raíz que EMR sin servidor pasa al maestro de aplicaciones de Tez.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMR sin servidor debería enviar los eventos ATS después de este período de tiempo tras la solicitud de cierre de AM.	0

Ajuste	Descripción	Valor predeterminado
`tez.am.speculation.enabled`	Opción que provoca el lanzamiento especulativo de tareas más lentas. Esto puede ayudar a reducir la latencia de los trabajos cuando algunas tareas se ejecutan más lentamente debido a máquinas defectuosas o lentas. Solo se admite con Amazon EMR 6.10.x y versiones anteriores.	`FALSE`
`tez.am.task.max.failed.attempts`	El número máximo de intentos que se pueden fallar para una tarea en particular antes de que la tarea de error. En este número no se incluyen 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 aleatorios de vértices. Esta característica se desactiva cuando el valor es 0. Las versiones 6.8.0 y posteriores de Amazon EMR admiten esta característica.	0
`tez.client.asynchronous-stop`	Opción que hace que EMR sin servidor envíe los eventos ATS antes de finalizar el controlador Hive.	`FALSE`

Ajuste	Descripción	Valor predeterminado
`tez.grouping.max-size`	El límite de tamaño superior (en bytes) de una división agrupada. Este límite evita que se produzcan divisiones 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
`tez.runtime.io.sort.mb`	El tamaño del búfer flexible cuando Tez ordena la salida.	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

Ajuste	Descripción	Valor predeterminado
`tez.shuffle-vertex-manager.max-src-fraction`	La fracción de tareas de la fuente que deben completarse antes de que EMR sin servidor programe todas las tareas para el vértice actual (en caso de una conexión `ScatterGather`). La cantidad de tareas listas para su programación en el vértice actual se escala linealmente entre `min-fraction` y `max-fraction`. Este valor predeterminado es el valor predeterminado o `tez.shuffle-vertex-manager.min-src-fraction`, el que sea mayor.	0.75
`tez.shuffle-vertex-manager.min-src-fraction`	La fracción de tareas de la fuente que deben completarse antes de que EMR sin servidor programe las tareas para el vértice actual (en caso de una conexión `ScatterGather`).	0,25
`tez.task.emr-serverless.launch.env.[`*KEY*`]`	Opción para establecer la variable de entorno *KEY* en el proceso de tareas de Tez. Para las tareas de Tez, este valor anula el valor `hive.emr-serverless.launch.env.[`*KEY*`]`.	
`tez.task.log.level`	El nivel de registro raíz que EMR sin servidor pasa al las tareas de Tez.	`INFO`

Ajuste	Descripción	Valor predeterminado
`tez.yarn.ats.event.flush.timeout.millis`	El tiempo máximo que AM debe esperar para que se vacíen los eventos antes de apagarse.	300000

Ejemplos de trabajos de Hive

El siguiente ejemplo de código muestra cómo ejecutar una consulta de Hive con la API `StartJobRun`.

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

Puede encontrar ejemplos adicionales de cómo ejecutar trabajos de Hive en el repositorio GitHub de Ejemplos de EMR sin servidor.

Resiliencia de trabajos de EMR sin servidor

Las versiones 7.1.0 y posteriores de EMR sin servidor incluyen compatibilidad para la resiliencia a trabajos, por lo que reintenta automáticamente cualquier trabajo fallido sin ninguna intervención manual por su parte. Otro beneficio de la resiliencia al trabajo es que EMR sin servidor traslada las ejecuciones de trabajos a diferentes zonas de disponibilidad (AZ) en caso de que una AZ experimente algún problema.

Para habilitar la resiliencia al trabajo en un trabajo, establezca la política de reintentos para su trabajo. Una política de reintentos garantiza que EMR sin servidor reinicie automáticamente un trabajo si se produce un error en algún momento. Las políticas de reintento se admiten para los trabajos por lotes y para los de streaming, por lo que puede personalizar la resiliencia a 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 al trabajo en trabajos en lotes y en streaming.

	Tareas por lotes	Trabajos de streaming
Comportamiento predeterminado	No vuelve a ejecutar el trabajo.	Siempre reintenta la ejecución del trabajo, ya que la aplicación crea puntos de comprobación mientras se ejecuta el trabajo.
Punto de reintento	Los trabajos por lotes no cuentan con puntos de comprobación, por lo que EMR sin servidor siempre reintenta la ejecución del trabajo desde su principio.	Los trabajos de streaming admiten puntos de comprobación, por lo que puede configurar la consulta de streaming para guardar el estado del tiempo de ejecución y el progreso hasta una ubicación de punto de comprobación en Amazon S3. EMR sin servidor reanuda la ejecución del trabajo desde el punto de comprobación. Para obtener más información, consulte Recovering from

	Tareas por lotes	Trabajos de streaming
		failures with Checkpointing en la documentación de Apache Spark.
Máximo de reintentos	Permite un máximo de 10 reintentos.	Los trabajos de streaming tienen un control de prevención de errores integrado, por lo que la aplicación deja de reintentar 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 puede personalizar el número máximo de intentos. Un valor de 1 indica que no hay reintentos.

Cuando EMR sin servidor reintenta la ejecución de un trabajo, también indexa el trabajo con un número de intentos, para que realice un seguimiento del ciclo de vida de un trabajo en todos sus intentos.

Use las operaciones de la API del EMR sin servidor o la AWS CLI para cambiar la resiliencia a los trabajos o acceder a información relacionada con la resiliencia a los trabajos. Para obtener más información, consulte la Guía de la API del EMR sin servidor.

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

En el siguiente ejemplo, se muestra cómo especificar un número máximo de 10 intentos al iniciar una ejecución de 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"
     }
}'
```

EMR sin servidor reintenta indefinidamente los trabajos de streaming en caso de fallar. Para evitar que se produzcan errores repetidos e irrecuperables, utilice el `maxFailedAttemptsPerHour` para configurar el control de prevención de errores para los reintentos de trabajos de streaming. Este parámetro le permite especificar el número máximo de intentos fallidos y permitidos una hora antes de que EMR sin servidor deje de volver a intentarlo. El valor predeterminado es cinco. Los valores aceptados son del 1 al 10, inclusive.

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

También puede usar las demás operaciones de la API de ejecución de trabajos para obtener información sobre los trabajos. Por ejemplo, use el parámetro `attempt` con la operación `GetJobRun` para obtener detalles sobre un intento de trabajo específico. Si no incluye el parámetro`attempt`, 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 operación `ListJobRunAttempts` 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 operación `GetDashboardForJobRun` crea y devuelve una URL que se usa para acceder a las IU de la aplicación para ejecutar un trabajo. El parámetro `attempt` le permite obtener una URL para un intento específico. Si no incluye el parámetro`attempt`, 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 de resiliencia al trabajo también agrega el nuevo evento Reintento de ejecución del trabajo de EMR sin servidor. EMR sin servidor publica este evento cada vez que se vuelve a intentar el trabajo. Puede utilizar esta notificación para realizar un seguimiento de los reintentos del trabajo. Para más información acerca de los eventos, consulte Eventos de Amazon EventBridge.

Registro con política de reintentos

Cada vez que EMR sin servidor reintenta un trabajo, el intento genera su propio conjunto de registros. Para garantizar que EMR sin servidor pueda entregar correctamente estos registros a Amazon S3 y Amazon CloudWatch sin sobrescribir ninguno, EMR sin servidor añade un prefijo al

formato de la ruta del registro de S3 y al nombre del flujo de registro de CloudWatch para incluir el número de intento del trabajo.

El siguiente es un ejemplo de cómo es su formato.

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

Este formato garantiza que EMR sin servidor publique todos los registros de cada intento de trabajo en su propia ubicación designada en Amazon S3 y CloudWatch. Para obtener más detalles, consulte Registros de almacenamiento.

> **Note**
>
> EMR sin servidor solo usa este formato de prefijo con todos los trabajos de streaming y cualquier trabajo por lotes que tengan habilitada la opción de reintento.

Configuración de metaalmacenes para EMR sin servidor

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 sin servidor, conserve los metadatos de esta tabla en un metaalmacén que tenga acceso a sus trabajos.

Dispone de dos opciones para un metaalmacén de Hive:

- El catálogo de datos de Glue de AWS
- Un metaalmacén de Apache Hive externo

Uso del catálogo de datos de Glue de AWS como metaalmacén

Puede configurar sus trabajos de Spark y Hive para que usen el catálogo de datos de Glue deAWS como su metaalmacén. Recomendamos esta configuración cuando se necesita un metaalmacén persistente o un metaalmacén compartido por diferentes servicios, aplicaciones o Cuentas de AWS. Para obtener más información acerca del catálogo de datos, consulte Cómo completar el Catálogo de datos de AWS Glue. Para obtener información acerca de los precios de AWS Glue, consulte Precios de AWS Glue.

Puede configurar su trabajo de EMR sin servidor para que utilizar el catálogo de datos de Glue de AWS en la misma aplicación de la Cuenta de AWS o en una Cuenta de AWS diferente.

Configuración del catálogo de datos de Glue de AWS

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

Spark

Cuando utiliza EMR Studio para ejecutar sus trabajos con aplicaciones EMR sin servidor Spark, el catálogo de datos de Glue de AWS es el metaalmacén predeterminado.

Cuando utilice los SDK o AWS CLI, establezca la configuración `spark.hadoop.hive.metastore.client.factory.class` en `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` en los parámetros `sparkSubmit` de la ejecución de su trabajo. En el siguiente ejemplo se muestra cómo configurar el catálogo de datos con la AWS CLI.

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

Como alternativa, puede establecer esta configuración al crear una nueva `SparkSession` en su 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

Para las aplicaciones EMR sin servidor Hive, el catálogo de datos es el metaalmacén predeterminado. Es decir, cuando ejecuta trabajos en una aplicación EMR sin servidor Hive, Hive registra la información del metaalmacén en el catálogo de datos en la misma Cuenta de AWS que 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 políticas de Glue de AWS necesarias que se describen en Configuración de los permisos de IAM para Glue de AWS.

Configuración del acceso entre cuentas para EMR sin servidor y el catálogo de datos de Glue de AWS

Para configurar el acceso entre cuentas para EMR sin servidor, primero inicie sesión en las siguientes Cuentas de AWS:

- `AccountA`: una Cuenta de AWS donde haya creado una aplicación EMR sin servidor.
- `AccountB`: una Cuenta de AWS que contenga un catálogo de datos de Glue de AWS al que desee que accedan sus ejecuciones de trabajos de EMR sin servidor.

1. Asegúrese de que un administrador u otra identidad autorizada en la `AccountB` adjunte una política de recursos al catálogo de datos en la `AccountB`. Esta política otorga permisos específicos entre cuentas de `AccountA` para realizar operaciones con los recursos del catálogo de `AccountB`.

   JSON

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

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

2. Agregue una política de IAM a rol de tiempo de ejecución del trabajo de EMR sin servidor en `AccountA` para que este rol pueda acceder a los recursos del catálogo de datos en `AccountB`.

   JSON

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

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

3. Inicio de su ejecución de trabajo. Este paso es ligeramente diferente según el tipo de aplicación EMR sin servidor de la `AccountA`.

   Spark

   Establezca la propiedad `spark.hadoop.hive.metastore.glue.catalogid` en `sparkSubmitParameters`, tal y como se muestra en el siguiente ejemplo. Sustituya *`AccountB-catalog-id`* por 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": {
        "entryPoint": "s3://amzn-s3-demo-bucket/scripts/test.py",
         "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.A
 --conf spark.hadoop.hive.metastore.glue.catalogid=AccountB-catalog-
id --conf spark.executor.cores=1 --conf spark.executor.memory=1g
 --conf spark.driver.cores=1 --conf spark.driver.memory=1g --conf
 spark.executor.instances=1"
    }
  }' \
--configuration-overrides '{
    "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
    "logUri": "s3://amzn-s3-demo-bucket/logs/"
```

```
    }
  }
}'
```

Hive

Establezca la propiedad `hive.metastore.glue.catalogid` en la clasificación `hive-site`, tal y como se muestra en el siguiente ejemplo. Sustituya *`AccountB-catalog-id`* por 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://amzn-s3-demo-bucket/hive/scripts/create_table.sql",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
hive/warehouse"
    }
}' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

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

Puede añadir archivos JAR auxiliares con `ADD JAR` en sus scripts de Hive. Para obtener información adicional, consulte Consideraciones a la hora de utilizar el catálogo de datos de AWS Glue.

Uso de un metaalmacén de Hive externo

Puede configurar sus trabajos de EMR sin servidor Spark y Hive para que se conecten a un metaalmacén de Hive externo, como Amazon Aurora o Amazon RDS para MySQL. En esta sección se describe cómo configurar un metaalmacén Hive de Amazon RDS, cómo configurar su VPC y cómo configurar sus trabajos de EMR sin servidor para usar un metaalmacén externo.

Creación de un metaalmacén de Hive externo

1. Cree una Amazon Virtual Private Cloud (Amazon VPC) con subredes privadas siguiendo las instrucciones de Creación de una VPC.
2. Cree su aplicación EMR sin servidor con su nueva Amazon VPC y sus subredes privadas. Cuando configura la aplicación EMR sin servidor con una VPC, aprovisiona en primer lugar una 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 acerca de cómo configurar una VPC, consulte Configuración del acceso a la VPC para que las aplicaciones EMR sin servidor se conecten a los datos.
3. Cómo crear una base de datos de MySQL o Aurora PostgreSQL en una subred privada de Amazon VPC. Para obtener información acerca de cómo crear una base de datos de Amazon RDS, consulte Creación de una instancia de base de datos de Amazon RDS.
4. Modifique el grupo de seguridad de la base de datos MySQL o Aurora para permitir las conexiones JDBC desde el grupo de seguridad del EMR sin servidor siguiendo los pasos que se indican en Modificación de una instancia de base de datos de Amazon RDS. Agregue una regla para el tráfico entrante al grupo de seguridad de RDS desde uno de sus grupos de seguridad del EMR sin servidor.

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

Configuración de las opciones de Spark

Uso de JDBC

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

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
```

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

El siguiente ejemplo de código es un script de punto de entrada de Spark que interactúa con un metaalmacén 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()
```

Uso del servicio Thrift

Puede configurar su aplicación EMR sin servidor Hive para que se conecte a un metaalmacén de Hive basado en una instancia Amazon RDS para MySQL o Amazon Aurora MySQL. Para ello, ejecute un servidor Thrift en el nodo principal de un clúster de Amazon EMR existente. Esta opción es ideal si ya tiene un clúster de Amazon EMR con un servidor Thrift que desea utilizar para simplificar las configuraciones de los trabajos de EMR sin servidor.

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

El siguiente ejemplo de código es un script de punto de entrada (`thriftscript.py`) que utiliza el protocolo Thrift para conectarse a un metaalmacén de Hive. Tenga en cuenta que la propiedad `hive.metastore.uris` 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()
```

Configuración de las opciones de Hive

Uso de JDBC

Si desea especificar una ubicación de base de datos de Hive externa en una instancia de Amazon RDS MySQL 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 comparte información del metaalmacén entre dos trabajos, debe asegurarse de no escribir en la misma tabla de metaalmacén simultáneamente, a menos que se escriba en distintas particiones de la misma tabla.

Establezca las siguientes configuraciones en la clasificación `hive-site` para activar el metaalmacén de Hive externo.

```
{
    "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 Thrift

Puede configurar su aplicación EMR sin servidor Hive para que se conecte a un metaalmacén de Hive basado en una instancia de Amazon RDS para MySQL o Amazon Aurora MySQL. Para ello, ejecute un servidor Thrift en el nodo principal de un clúster de Amazon EMR existente. Esta opción es ideal si ya tiene un clúster de Amazon EMR que ejecuta un servidor Thrift y desea utilizar las configuraciones de trabajo de EMR sin servidor.

Establezca las siguientes configuraciones en la clasificación `hive-site` para que EMR sin servidor pueda acceder al metaalmacén remoto de Thrift. Tenga en cuenta que la propiedad `hive.metastore.uris` debe establecerse 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"
    }
}
```

Cómo trabajar con la jerarquía de catálogos múltiples de AWS Glue en EMR sin servidor

Puede configurar sus aplicaciones de EMR sin servidor para que funcionen con la jerarquía de catálogos múltiples de AWS Glue. El siguiente ejemplo muestra cómo usar EMR-S Spark con la jerarquía de catálogos múltiples de AWS Glue.

Para obtener más información sobre la jerarquía de catálogos múltiples, consulte Working with a multi-catalog hierarchy in AWS Glue Data Catalog with Spark on Amazon EMR.

Uso de Almacenamiento administrado de Redshift (RMS) con Iceberg y el Catálogo de datos de AWS Glue

A continuación, se muestra cómo configurar Spark para su integración con el Catálogo de datos de AWS Glue y Iceberg:

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

Un ejemplo de consulta de una tabla del catálogo, tras la integración:

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

Uso de Almacenamiento administrado de Redshift (RMS) con la API REST de Iceberg y el Catálogo de datos de AWS Glue

A continuación, se muestra cómo configurar Spark para que funcione con el catálogo REST de Iceberg:

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

```
    --conf spark.sql.catalog.rms.warehouse=Glue RMS catalog ID
    --conf spark.sql.catalog.rms.uri=Glue endpoint URI/iceberg
    --conf spark.sql.catalog.rms.rest.sigv4-enabled=true
    --conf spark.sql.catalog.rms.rest.signing-name=glue
    --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
    }
 }'
```

Un ejemplo de consulta de una tabla del catálogo:

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

Consideraciones sobre el uso de un metaalmacén externo

- Puede configurar bases de datos que sean compatibles con MariaDB JDBC como su metaalmacén. Algunos ejemplos de estas bases de datos son RDS para MariaDB, MySQL y Amazon Aurora.
- Los metaalmacenes no se inicializan de forma automática. Si su metaalmacén no se ha inicializado con un esquema para su versión de Hive, utilice la Herramienta de esquemas de Hive.
- EMR sin servidor no admite la autenticación de Kerberos. No puede utilizar un servidor de metaalmacén Thrift con autenticación de Kerberos con trabajos EMR sin servidor Spark o Hive.
- Debe configurar el acceso a la VPC para usar la jerarquía de catálogos múltiples.

Acceso a los datos de S3 en otra cuenta de AWS desde EMR sin servidor

Puede ejecutar trabajos de Amazon EMR sin servidor desde una cuenta AWS y configurarlos para acceder a los datos de los buckets de Amazon S3 que pertenezcan a otra cuenta AWS. En esta página se describe cómo configurar el acceso entre cuentas a S3 desde EMR sin servidor.

Los trabajos que se ejecutan en EMR sin servidor pueden utilizar una política de bucket de S3 o un rol asumido para acceder a los datos de Amazon S3 desde una cuenta AWS diferente.

Requisitos previos

Para configurar el acceso entre cuentas para Amazon EMR sin servidor, complete las tareas mientras tenga la sesión iniciada en dos cuentas de AWS:

- **`AccountA`**: esta es la cuenta de AWS en la que ha creado una aplicación Amazon EMR sin servidor. Antes de configurar el acceso entre cuentas tenga preparado en su cuenta lo siguiente:
  - Una aplicación de Amazon EMR sin servidor en la que desee ejecutar los trabajos.
  - Un rol de ejecución de trabajos que tenga los permisos necesarios para ejecutar trabajos en la aplicación. Para obtener más información, consulta Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.
- **`AccountB`**: esta es la cuenta de AWS que contiene el bucket de S3 al que desea que accedan sus trabajos de Amazon EMR sin servidor.

Uso de una política de buckets de S3 para acceder a los datos de S3 entre cuentas

Para acceder al bucket de S3 en account B desde account A, adjunte la siguiente política al bucket de S3 en account B.

JSON

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

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

Para obtener más información acerca del acceso entre cuentas de S3 con políticas de bucket de S3, consulte Ejemplo 2: propietario del bucket que concede permisos de bucket entre cuentas en la Guía del usuario de Amazon Simple Storage Service.

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

Otra forma de configurar el acceso entre cuentas para Amazon EMR sin servidor es con la acción `AssumeRole` desde 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 llamadas de API a EMR sin servidor y Amazon S3 con las credenciales de seguridad temporales que usted crea con `AssumeRole`.

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

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 rol de IAM `Cross-Account-Role-B` en `AccountB` que pueda acceder al *`cross-account-bucket`*.
   a. Inicie sesión en Consola de administración de AWS y abra la consola IAM 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 acerca de cómo crear un rol de IAM, consulte Creación de roles de IAM en la Guía del usuario de IAM.
   c. Cree una política de IAM que especifique los permisos del `Cross-Account-Role-B` para acceder al bucket de S3 de *`cross-account-bucket`*, tal como se muestra en la siguiente instrucción de política. Adjunte la política de IAM al `Cross-Account-Role-B`. Para obtener más información, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::cross-account-bucket",
        "arn:aws:s3:::cross-account-bucket/*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

Si requiere acceso a DynamoDB, cree una política de IAM que especifique los permisos para acceder a la tabla de DynamoDB entre cuentas. Adjunte la política de IAM al `Cross-Account-Role-B`. Para obtener más información, consulte Amazon DynamoDB: permite el acceso a una determinada tabla en la Guía del usuario de IAM.

A continuación, se presenta una política para acceder a una tabla de DynamoDB `CrossAccountTable`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:*"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:123456789012:table/CrossAccountTable"
      ],
      "Sid": "AllowDYNAMODB"
    }
```

```
  ]
}
```

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

      JSON

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

4. Conceda a `Job-Execution-Role-A` en `AccountA` el permiso AWS STS `AssumeRole` para asumir `Cross-Account-Role-B`.
   a. En la consola de IAM de la cuenta de AWS `AccountA`, seleccione `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`.

      JSON

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

```
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/Cross-Account-Role-B"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

Ejemplos de roles asumidos

Use un solo rol asumido para acceder a todos los recursos de S3 de una cuenta o, con Amazon EMR 6.11 y versiones posteriores, configure varios roles de IAM para asumirlos al acceder a diferentes buckets de S3 entre cuentas.

Temas

- Acceso a los recursos de S3 con un rol asumido
- Acceso a los recursos de S3 con varios roles asumidos

Acceso a los recursos de S3 con un rol asumido

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

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

3. Para Hive, utilice `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, `tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` y `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` para especificar 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 una ejecución de un trabajo EMR sin servidor con acceso entre cuentas.

Spark

El siguiente ejemplo muestra cómo usar un rol asumido para iniciar la ejecución de un trabajo de EMR sin servidor Spark con acceso entre cuentas 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":
 "com.amazonaws.emr.AssumeRoleAWSCredentialsProvider",
                "spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Hive

El siguiente ejemplo muestra cómo usar un rol asumido para iniciar la ejecución de un trabajo de EMR sin servidor Hive con acceso entre cuentas 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"
            }
        }]
    }'
```

Acceso a los recursos de S3 con varios roles asumidos

Con las versiones 6.11.0 y posteriores de EMR sin servidor, configure varios roles de IAM para que los asuma al acceder a diferentes buckets entre cuentas. Si desea acceder a diferentes recursos de S3 con diferentes roles asumidos en la cuenta B, utilice las siguientes configuraciones al iniciar la ejecución del trabajo:

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

2. Especifique la configuración `fs.s3.bucketLevelAssumeRoleMapping` de EMRFS para definir la asignación desde el nombre del bucket de S3 hasta el rol de IAM que se va a asumir en la cuenta B. El formato del valor debe ser `bucket1->role1;bucket2->role2`.

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

Spark

El siguiente ejemplo muestra cómo usar varios roles asumidos para crear la ejecución de un trabajo de EMR sin servidor 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 la ejecución de un trabajo de EMR sin servidor 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 sin servidor

Use la siguiente información para diagnosticar y resolver los problemas habituales que ocurren cuando se trabaja con Amazon EMR sin servidor.

Temas

- Error: el trabajo ha fallado porque la cuenta ha alcanzado el límite de servicio en la cantidad máxima de vCPU que puede utilizar simultáneamente.
- Error: el trabajo falló porque la aplicación superó la configuración de capacidad máxima.
- Error: el trabajo falló debido a que no se pudo asignar el trabajador porque la aplicación ha superado la capacidad máxima.

- Error: Acceso de S3 denegado. 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. Consulte la guía del usuario sobre cómo utilizar las bibliotecas de Python con EMR sin servidor.
- 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: el trabajo ha fallado porque la cuenta ha alcanzado el límite de servicio en la cantidad máxima de vCPU que puede utilizar simultáneamente.

Este error indica que EMR sin servidor no pudo enviar el trabajo porque la cuenta ha superado la capacidad máxima. Aumente la capacidad máxima de la cuenta. Compruebe sus límites de servicio en las cuotas de servicio de EMR Serverless.

Error: el trabajo falló porque la aplicación superó la configuración de capacidad máxima.

Este error indica que EMR sin servidor no pudo enviar el trabajo porque la aplicación ha superado la capacidad máxima configurada. Aumente la capacidad máxima de la aplicación.

Error: el trabajo falló debido a que no se pudo asignar el trabajador porque la aplicación ha superado la capacidad máxima.

Este error indica que el trabajo no se pudo completar. No se pudieron asignar los trabajadores porque la aplicación superó la configuración de capacidad máxima.

Error: Acceso de S3 denegado. 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 acerca de los roles de tiempos de ejecución, consulte Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.

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

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, consulte Uso de bibliotecas de Python con EMR sin servidor.

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 sin servidor. Para comprobar que el rol de IAM existe y validar que ha configurado correctamente la política de confianza del rol, consulte las instrucciones que aparecen en Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor.

Ejecutar cargas de trabajo interactivas con EMR sin servidor a través de EMR Studio

Con las aplicaciones interactivas EMR sin servidor, puede ejecutar cargas de trabajo interactivas para Spark con EMR sin servidor mediante cuadernos alojados en EMR Studio.

Descripción general

Una aplicación interactiva es una aplicación EMR sin servidor que tiene capacidades interactivas habilitadas. Con las aplicaciones interactivas Amazon EMR sin servidor, puede ejecutar cargas de trabajo interactivas con los cuadernos de Jupyter que se administran en Amazon EMR Studio. 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 sin servidor incluyen los siguientes:

- Los ingenieros de datos utilizan la experiencia de IDE en EMR Studio para crear un script de ETL. El script incorpora datos de las instalaciones, los transforma para su análisis y los almacena en Amazon S3.
- Los científicos de datos usan cuadernos para explorar los conjuntos de datos y entrenan modelos de machine learning (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 sin servidor, debe cumplir con los siguientes requisitos:

- Las aplicaciones interactivas de EMR sin servidor son compatibles con la versión 6.14.0 y posteriores de Amazon EMR.
- Para acceder a su aplicación interactiva, ejecute las cargas de trabajo que envíe y ejecute cuadernos interactivos desde EMR Studio, necesita permisos y roles específicos. Para obtener más información, consulta 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 sin servidor, debe configurar permisos adicionales para su identidad o rol de IAM:

Para acceder a su aplicación interactiva

Configure los permisos de usuario y de espacio de trabajo para EMR Studio. Para obtener más información, consulte Configure EMR Studio user permissions en la Guía de administración de Amazon EMR.

Para ejecutar las cargas de trabajo que envíe con EMR sin servidor

Configure un rol de tiempo de ejecución. Para obtener más información, consulta Crear un rol de tiempo de ejecución del trabajo.

Para ejecutar los cuadernos interactivos desde EMR Studio

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

- **`emr-serverless:AccessInteractiveEndpoints`** - Otorga permiso para acceder y conectarse a la aplicación interactiva que usted especifique como `Resource`. Este permiso es necesario para conectarse a una aplicación de EMR sin servidor desde un espacio de trabajo de EMR Studio.
- **`iam:PassRole`** - Otorga permiso para acceder al rol de ejecución de IAM que usted tiene previsto utilizar al adjuntarlo a una aplicación. El permiso `PassRole` apropiado es necesario para conectarse a una aplicación de EMR sin servidor desde un espacio de trabajo de EMR Studio.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
```

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

Configuración de aplicaciones interactivas

Siga los siguientes pasos de alto nivel para crear una aplicación EMR sin servidor con capacidades interactivas de Amazon EMR Studio en la Consola de administración de AWS.

1. Siga los pasos en la Introducción a Amazon EMR sin servidor para crear una aplicación.
2. A continuación, inicie un espacio de trabajo desde EMR Studio y asócielo a una aplicación EMR sin servidor como opción de computación. Para obtener más información, consulte la pestaña Interactive workload en el paso 2 de la documentación de introducción a EMR sin servidor.

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

Consideraciones sobre las aplicaciones interactivas

- Las aplicaciones interactivas de EMR sin servidor son compatibles con la versión 6.14.0 y posteriores de Amazon EMR.

- EMR Studio es el único cliente que está integrado con las aplicaciones interactivas EMR sin servidor. Las siguientes capacidades de EMR Studio no son compatibles con las aplicaciones interactivas EMR sin servidor: colaboración en el espacio de trabajo, SQL Explorer y ejecución programática de cuadernos.
- Las aplicaciones interactivas solo son compatibles con el motor Spark.
- Las aplicaciones interactivas son compatibles con los kernels de Python 3, PySpark y Spark Scala.
- Puede ejecutar hasta 25 cuadernos simultáneos en una sola aplicación interactiva.
- No hay un punto de conexión ni una interfaz de API que admita los cuadernos Jupyter autoalojados con aplicaciones interactivas.
- Para una experiencia de inicio optimizada, le sugerimos que configure la capacidad preinicializada para los controladores y ejecutores y que inicie previamente la aplicación. Al iniciar previamente la aplicación, asegúrese de que esté lista cuando quiera asociarla a su 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 solicitud de `create-application` o de `update-application`.
- Cuando utilice una aplicación interactiva, le sugerimos que configure una capacidad preinicializada de kernels, controladores y ejecutores para ejecutar sus cuadernos. Cada sesión interactiva de Spark requiere un kernel y un controlador, por lo que EMR sin servidor mantiene un trabajador de kernel preinicializado para cada controlador preinicializado. De forma predeterminada, EMR sin servidor mantiene la capacidad preinicializada de un trabajador de kernel en toda la aplicación, incluso si no se especifica ninguna capacidad preinicializada para los controladores. Cada trabajador de kernel usa 4 vCPU y 16 GB de memoria. Para obtener información actual acerca de los precios, consulte la página Precios de Amazon EMR.
- Debe tener una cuota de servicio de vCPU suficiente en su Cuenta de AWS para ejecutar cargas de trabajo interactivas. Si no ejecuta cargas de trabajo compatibles con Lake Formation, le sugerimos tener al menos 24 vCPU. De lo contrario, le sugerimos contar al menos con 28 vCPU.
- EMR sin servidor finaliza automáticamente los kernels de los cuadernos si han estado inactivos durante más de 60 minutos. EMR sin servidor calcula el tiempo de inactividad del kernel a partir de la última actividad completada durante la sesión del cuaderno. Actualmente, no puede modificar la configuración del tiempo de espera de inactividad del kernel.

- Para habilitar Lake Formation con cargas de trabajo interactivas, establezca la configuración `spark.emr-serverless.lakeformation.enabled` en `true` bajo la clasificación `spark-defaults` en el objeto `runtime-configuration` al crear una aplicación EMR sin servidor. Para obtener más información, consulte Cómo activar Lake Formation en Amazon EMR.

Ejecutar cargas de trabajo interactivas con EMR sin servidor a través de un punto de conexión Apache Livy

Con Amazon EMR 6.14.0 y versiones posteriores, puede crear y habilitar un punto de conexión Apache Livy mientras crea una aplicación EMR sin servidor y ejecutar cargas de trabajo interactivas a través de sus cuadernos autoalojados o con un cliente personalizado. Un punto de conexión Apache Livy ofrece los siguientes beneficios:

- Puede conectarse de forma segura a un punto de conexión Apache Livy a través de los cuadernos de Jupyter y administrar las cargas de trabajo de Apache Spark con la interfaz REST de Apache Livy.
- Utilice las operaciones de la API de REST de Apache Livy para aplicaciones web interactivas que utilizan datos de las cargas de trabajo de Apache Spark.

Requisitos previos

Para utilizar un punto de conexión de Apache Livy con EMR sin servidor, debe cumplir con los siguientes requisitos:

- Complete los pasos que se indican en Introducción a Amazon EMR sin servidor.
- Para ejecutar cargas de trabajo interactivas a través de los puntos de conexión de Apache Livy, necesita ciertos permisos y roles. Para más información, consulte Required permissions for interactive workloads.

Permisos necesarios

Además de los permisos necesarios para acceder a EMR sin servidor, también debe agregar los siguientes permisos a su rol de IAM para acceder a un punto de conexión de Apache Livy y ejecutar aplicaciones:

- `emr-serverless:AccessLivyEndpoints` - otorga permiso para acceder y conectarse a la aplicación compatible con Livy que usted especifica como `Resource`. Necesita este permiso para ejecutar las operaciones de la API de REST disponibles en el punto de conexión de Apache Livy.
- `iam:PassRole`: concede permiso para acceder al rol de ejecución de IAM cuando crea la sesión de Apache Livy. EMR sin servidor utilizará este rol para ejecutar sus cargas de trabajo.
- `emr-serverless:GetDashboardForJobRun`— concede permiso para generar la IU de Spark Live y los enlaces al registro de controladores y, además, proporciona acceso a los registros como parte de los resultados de la sesión de Apache Livy.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessLivyEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
    },
    {
```

```
      "Sid": "EMRServerlessDashboardAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetDashboardForJobRun"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    }
  ]
}
```

Introducción

Para crear una aplicación compatible con Apache Livy y ejecutarla, siga estos pasos.

1. Para crear una aplicación compatible con Apache Livy y ejecutarla, siga este comando.

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

2. Después de que EMR sin servidor haya creado su aplicación, inicie la aplicación para que el punto de conexión de Apache Livy esté disponible.

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

   Utilice el siguiente comando para comprobar el estado de la aplicación. Una vez que el estado pase a `STARTED`, acceda al punto de conexión de Apache Livy.

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

3. Utilice la siguiente URL para acceder al punto de conexión:

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

Una vez que el punto de conexión esté listo, envíe cargas de trabajo en función de su caso de uso. Debe firmar todas las solicitudes que se envíen al punto de conexión con el protocolo SigV4 y pasar un encabezado de autorización. Puede usar uno de los métodos siguientes para ejecutar cargas de trabajo:

- Cliente de HTTP: envíe sus operaciones de la API de punto de conexión de Apache Livy con un cliente de HTTP personalizado.
- Kernel de Sparkmagic: ejecute el kernel de Sparkmagic de forma local y enviar consultas interactivas con los cuadernos de Jupyter.

Clientes de HTTP

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

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

La tabla siguiente describe todas las operaciones de la API de Apache Livy disponibles.

Operación de la API	Descripción
GET /sessions	Devuelve una lista de todas las sesiones interactivas activas.
POST /sessions	Crea una nueva sesión interactiva mediante Spark o Pyspark.
GET /sessions/<*sessionId* >	Devuelve la información de la sesión.
GET /sessions/<*sessionId* >/state	Devuelve el estado de la sesión.
DELETE /sessions/<*sessionId* >	Detiene y elimina la sesión.

Operación de la API	Descripción
GET /sessions/<*sessionId* >/statements	Devuelve todas las instrucciones de una sesión.
POST /sessions/<*sessionId* >/statements	Ejecuta una instrucción en una sesión.
GET /sessions/<*sessionId* >/statements/<*statementId* >	Devuelve los detalles de la instrucción especificada en una sesión.
POST /sessions/<*sessionId* >/statements/<*statementId* >/cancel	Cancela la instrucción especificada en esta sesión.

Envío de solicitudes al punto de conexión de Apache Livy

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

Antes de empezar a enviar solicitudes al punto de conexión, asegúrese de haber instalado las siguientes bibliotecas:

```
pip3 install botocore awscrt requests
```

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

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
import botocore.session
import json, pprint, textwrap

endpoint = 'https://<application_id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com'
headers = {'Content-Type': 'application/json'}

session = botocore.session.Session()
```

```
signer = crt.auth.CrtS3SigV4Auth(session.get_credentials(), 'emr-serverless',
 '<AWS_REGION>')


### Create session request

data = {'kind': 'pyspark', 'heartbeatTimeoutInSecond': 60, 'conf': { 'emr-
serverless.session.executionRoleArn': 'arn:aws:iam::123456789012:role/role1'}}

request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r.json())


### List Sessions Request

request = AWSRequest(method='GET', url=endpoint + "/sessions", headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r2 = requests.get(prepped.url, headers=prepped.headers)
pprint.pprint(r2.json())


### Get session state

session_url = endpoint + r.headers['location']

request = AWSRequest(method='GET', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False
```

```
signer.add_auth(request)

prepped = request.prepare()

r3 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r3.json())


### Submit Statement

data = {
      'code': "1 + 1"
}

statements_url = endpoint + r.headers['location'] + "/statements"

request = AWSRequest(method='POST', url=statements_url, data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r4 = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r4.json())

### Check statements results

specific_statement_url = endpoint + r4.headers['location']

request = AWSRequest(method='GET', url=specific_statement_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r5 = requests.get(prepped.url, headers=prepped.headers)
```

```
pprint.pprint(r5.json())


### Delete session


session_url = endpoint + r.headers['location']

request = AWSRequest(method='DELETE', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r6 = requests.delete(prepped.url, headers=prepped.headers)

pprint.pprint(r6.json())
```

Kernel de Sparkmagic

Antes de instalar Sparkmagic, asegúrese de haber configurado las AWS credenciales en la instancia en la que quiere instalar Sparkmagic

1. Instale Sparkmagic siguiendo los pasos de instalación. Tenga en cuenta que solo debe realizar los primeros cuatro pasos.
2. El kernel de Sparkmagic admite autenticadores personalizados, por lo que puede integrar un autenticador con el núcleo de Sparkmagic para que todas las solicitudes estén firmadas mediante SigV4.
3. Instalar el autenticador personalizado EMR sin servidor.

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

4. Ahora proporciona la ruta al autenticador personalizado y la URL del punto de conexión de Apache Livy en el archivo JSON de configuración de Sparkmagic. Utilice el comando siguiente para abrir el archivo de configuración.

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

A continuación se muestra un archivo `config.json` de ejemplo.

```
{
"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 el laboratorio de Jupyter. Debe usar la autenticación personalizada que configuró en el último paso.
6. A continuación, puede ejecutar los siguientes comandos del cuaderno 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 operaciones de la API de Apache Livy a través de la URL del punto de conexión de Apache Livy configurada. A continuación, puede escribir sus 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 de conexión de Apache Livy.

- EMR sin servidor mantiene el aislamiento a nivel de sesión mediante la entidad principal que llama. La entidad principal que llama y crea la sesión es la única que puede acceder a esa sesión. Para un aislamiento más detallado, configure una identidad de fuente al asumir las credenciales. En este caso, EMR sin servidor impone el aislamiento a nivel de sesión en función de la entidad principal que llama y de la identidad de fuente. Para obtener más información sobre la identidad de fuente, consulte Monitorear y controlar las acciones realizadas con roles asumidos.
- Los puntos de conexión de Apache Livy son compatibles con EMR sin servidor 6.14.0 y versiones posteriores.
- Los puntos de conexión de Apache Livy solo son compatibles con el motor Apache Spark.
- Los puntos de conexión de Apache Livy son compatibles con Scala Spark y PySpark.
- De forma predeterminada, `autoStopConfig` está habilitada en las aplicaciones. Esto significa que las aplicaciones se cierran después de 15 minutos de estar inactivas. Puede cambiar esta configuración como parte de su solicitud de `create-application` o de `update-application`.
- Puede ejecutar hasta 25 sesiones simultáneas en una sola aplicación habilitada para puntos de conexión de Apache Livy.
- Para una experiencia de inicio optimizada, le sugerimos que configure la capacidad preinicializada para los controladores y ejecutores.
- Debe iniciar la aplicación manualmente antes de conectarse al punto de conexión de Apache Livy.

- Debe tener una cuota de servicio de vCPU suficiente en su Cuenta de AWS para ejecutar cargas de trabajo interactivas. Se recomiendan al menos 24 vCPU.
- El tiempo de espera predeterminado de una sesión de Apache Livy es de 1 hora. Si no ejecuta las instrucciones durante una hora, entonces, Apache Livy eliminará la sesión y liberará el controlador y los ejecutores. A partir de la versión emr-7.8.0, este valor se puede establecer mediante la especificación del parámetro `ttl` como parte de la solicitud `/sessions POST` de Livy, por ejemplo, 2h (horas), `120m` (minutos), `7200s` (segundos), `7200000ms` (milisegundos).

> **Note**
>
> Esta configuración no se puede modificar en las versiones anteriores a emr-7.8.0. El siguiente ejemplo es una muestra del cuerpo de una solicitud `POST /sessions`.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "executionRoleArn"
    },
    "ttl": "2h"
}
```

- A partir de la versión emr-7.8.0 de Amazon EMR para aplicaciones con control de acceso detallado a través de LakeFormation, la configuración se puede deshabilitar por sesión. Para obtener más información sobre cómo habilitar un control de acceso detallado para una aplicación de EMR sin servidor, consulte Methods for fine-grained access control.

> **Note**
>
> No es posible habilitar Lake Formation para una sesión si no se habilitó para una aplicación. El siguiente ejemplo es una muestra del cuerpo de una solicitud `POST /sessions`.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
```

```
    "conf": {
        "emr-serverless.session.executionRoleArn": "executionRoleArn"
    },
    "spark.emr-serverless.lakeformation.enabled" : "false"
}
```

- Solo las sesiones activas pueden interactuar con un punto de conexión de Apache Livy. Una vez que la sesión finalice, se cancele o termine, no podrá acceder a ella a través del punto de conexión de Apache Livy.

Registro y supervisión

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de los trabajos y las aplicaciones del EMR sin servidor. Debe recopilar datos de monitorización de todas las partes de su solución de EMR sin servidor para que le resulte más sencillo depurar un error que se produce en distintas partes del código, en caso de que ocurra.

Temas

Almacenamiento de registros

Para monitorizar el progreso de su trabajo en EMR sin servidor y solucionar errores en los trabajos, elija la forma en que EMR sin servidor almacena y proporciona los registros de las aplicaciones. Cuando envíe una ejecución de trabajo, especifique el almacenamiento administrado, Amazon S3 y Amazon CloudWatch como opciones de registro.

Con CloudWatch, especifique los tipos de registro y las ubicaciones de registro que quiere usar o aceptar los tipos y ubicaciones predeterminados. Para obtener más información sobre los registros de CloudWatch, consulte the section called “Amazon CloudWatch”. Con el almacenamiento gestionado y el registro de S3, en la siguiente tabla se muestran las ubicaciones de registro y la disponibilidad de la IU que puede esperar si elige almacenamiento administrado, buckets de Amazon S3 o ambos.

Opción	Registros de eventos	Registros de contenedor	IU de las aplicaciones
Almacenamiento administrado	Almacenado en un almacenamiento administrado	Almacenado en un almacenamiento administrado	Compatible

Opción	Registros de eventos	Registros de contenedor	IU de las aplicaciones
Tanto almacenamiento administrado como bucket de S3	Almacenado en ambos lugares	Almacenado en un bucket de S3	Compatible
Bucket de Amazon S3	Almacenado en un bucket de S3	Almacenado en un bucket de S3	No admitido[1]

[1] Le sugerimos que mantenga seleccionada la opción de almacenamiento administrado. De lo contrario, no podrá utilizar las IU de la aplicación integradas.

Registro para EMR sin servidor con almacenamiento administrado

De forma predeterminada, EMR sin servidor almacena los registros de las aplicaciones de forma segura en el almacenamiento administrado de Amazon EMR durante un máximo de 30 días.

> **Note**
>
> Si desactiva la opción predeterminada, Amazon EMR no podrá solucionar los problemas de sus trabajos en su nombre. Ejemplo: No puede acceder a la IU de Spark desde la consola sin servidor de EMR.

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

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

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

Si su aplicación de EMR sin servidor se encuentra en una subred privada con puntos de conexión de VPC para Amazon S3 y adjunta una política de puntos de conexión para controlar el acceso, agregue los siguientes permisos para que EMR sin servidor almacene y proporcione los registros de la aplicación. Reemplace `Resource` por los buckets `AppInfo` de la tabla de regiones disponible en Sample policies for private subnets that access Amazon S3.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessManagedLogging",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::prod.us-east-1.appinfo.src",
        "arn:aws:s3:::prod.us-east-1.appinfo.src/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalServiceName": "emr-serverless.amazonaws.com",
          "aws:SourceVpc": "vpc-12345678"
        }
      }
    }
  ]
}
```

Además, utilice la clave de condición `aws:SourceVpc` para asegurarse de que la solicitud pase por la VPC a la que está asociado el punto de conexión de VPC.

Registro de EMR sin servidor con buckets de Amazon S3

Para que sus trabajos puedan enviar datos de registro a Amazon S3, debe incluir los siguientes permisos en la política de permisos del rol de tiempo de ejecución del trabajo. Reemplace *amzn-s3-demo-logging-bucket* con el nombre de su bucket de registro.

JSON

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

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

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/"
        }
    }
}
```

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

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

EMR sin servidor almacena los registros del controlador Spark en la siguiente ruta

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

EMR sin servidor almacena los registros del ejecutor Spark en la siguiente ruta

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_EXECUTOR/<EXECUTOR-ID>'
```

<EXECUTOR-ID> es un número entero.

Las versiones 7.1.0 y posteriores de EMR sin servidor admiten reintentos para trabajos de streaming y trabajos por lotes. Si ejecuta un trabajo con los reintentos habilitados, EMR sin servidor agrega automáticamente un número de intento al prefijo de la ruta de registro, para poder hacer una mejor distinción y seguimiento de los registros.

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

Registro de EMR sin servidor con Amazon CloudWatch

Cuando envíe un trabajo a una aplicación EMR sin servidor, elija Amazon CloudWatch como opción para almacenar los registros de la aplicación. Esto le permite utilizar las funciones de análisis de registros de CloudWatch, como CloudWatch Logs Insights y Live Tail. También puede transmitir registros desde CloudWatch a otros sistemas, como OpenSearch, para analizarlos más a fondo.

EMR sin servidor proporciona registro en tiempo real de los registros de los controladores. Puede acceder a los registros en tiempo real con la función de seguimiento en directo de CloudWatch o mediante los comandos de seguimiento de la CLI de CloudWatch.

De forma predeterminada, el registro de CloudWatch está deshabilitado para EMR sin servidor. Para habilitarlo, utilice la configuración de AWS CLI.

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

Permisos necesarios para realizar un registro con CloudWatch

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

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:*"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSPutlogevents"
    }
  ]
}
```

AWS CLI

Para configurar Amazon CloudWatch para almacenar los registros de EMR sin servidor de la AWS CLI, utilice la configuración de `cloudWatchLoggingConfiguration` al iniciar la ejecución de un trabajo. Para ello, incluya las siguientes anulaciones de configuración. Opcionalmente, también

puede proporcionar un nombre de grupo de registro, 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 de registros predeterminado `/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`*.

Las versiones 7.1.0 y posteriores de EMR sin servidor admiten reintentos para trabajos de streaming y trabajos por lotes. Si habilitó reintentos para un trabajo, EMR sin servidor agrega automáticamente un número de intento al prefijo de la ruta de registro, para poder hacer una mejor distinción y seguimiento de los registros.

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

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

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

El siguiente ejemplo muestra todas las configuraciones obligatorias y opcionales que puede especificar al activar el registro de Amazon CloudWatch para EMR sin servidor. Los valores `logTypes` admitidos también se muestran en el siguiente 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 sin servidor publica solo los registros stdout y stderr del controlador en CloudWatch. Si desea obtener otros registros, especifique el rol de contenedor y los tipos de registro correspondientes con el campo `logTypes`.

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

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 sin servidor puede rotar los registros de aplicaciones y los registros de eventos de Spark. La rotación de registros ayuda a solucionar el problema de que los trabajos de larga duración 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, debido a que no quede 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 de lanzamiento de Amazon EMR.

En lugar de generar un único archivo de registro de eventos, EMR sin servidor rota el registro de eventos en un intervalo de tiempo periódico y elimina los archivos de registro de eventos anteriores. 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 registros de eventos de Spark está disponible en todas las etiquetas de lanzamiento de Amazon EMR.

EMR sin servidor también rota los registros de aplicaciones de Spark para los controladores y ejecutores, como los archivos `stdout` y `stderr`. Para acceder a los archivos de registro más recientes, seleccione 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 consultar los registros rotados anteriores, especifique una ubicación de Amazon S3 cuando almacene 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. EMR sin servidor 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 ocupa 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.

Las versiones 7.1.0 y posteriores de EMR sin servidor admiten reintentos para trabajos de streaming y por lotes. Si habilitó los reintentos en su tarea, EMR sin servidor añade un prefijo a la ruta de los registros de dichos trabajos para que pueda hacerse una mejor distinción y seguimiento de los registros entre sí. Esta ruta contiene todos los registros rotados.

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

Cifrado de registros

Cifrado de registros de EMR sin servidor con almacenamiento administrado

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

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

Cifrado de registros de EMR sin servidor con buckets de Amazon S3

Para cifrar los registros en su bucket de Amazon S3 con su propia clave KMS, utilice la configuración `s3MonitoringConfiguration` al enviar una ejecución de trabajos.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/",
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Cifrado de registros de EMR sin servidor con Amazon CloudWatch

Para cifrar los registros en Amazon CloudWatch con su propia clave KMS, utilice la configuración `cloudWatchLoggingConfiguration` al enviar una ejecución de trabajos.

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

```
        }
    }
}
```

Permisos necesarios para el cifrado de registros

En esta sección

Permisos de usuario necesarios

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

Política de claves de ejemplo

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

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

Política de IAM de ejemplo

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

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Para iniciar la IU de Spark o Tez, conceda a sus usuarios, grupos o roles los siguientes permisos de acceso a la API `emr-serverless:GetDashboardForJobRun` de la siguiente manera:

JSON

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

Permisos de clave de cifrado para Amazon S3 y almacenamiento administrado

Cuando cifra los registros con su propia clave de cifrado en el almacenamiento administrado o en los buckets de S3, debe configurar los permisos de clave de KMS de la siguiente manera.

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

```
{
    "Effect": "Allow",
    "Principal":{
       "Service": "emr-serverless.amazonaws.com"
     },
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey"
      ],
     "Resource": "*"
     "Condition": {
       "StringLike": {
         "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
       }
     }
 }
```

Como práctica recomendada de seguridad, le sugerimos que agregue una clave de condición `aws:SourceArn` a la política de claves de KMS. La clave de condición global de IAM `aws:SourceArn` ayuda a garantizar que EMR sin servidor utilice la clave de KMS solo para un ARN de aplicación.

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

JSON

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

```
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Permisos de clave de cifrado para Amazon CloudWatch

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

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:AssociateKmsKey"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSAssociatekmskey"
    }
  ]
}
```

Configure la política de claves de KMS para conceder permisos de KMS a Amazon CloudWatch:

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:*:123456789012:*"
        }
      },
      "Sid": "AllowKMSDecrypt"
    }
  ]
}
```

Configure las propiedades de Apache Log4j2 para Amazon EMR sin servidor

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

Configure las propiedades de Spark Log4j2 para Amazon EMR sin servidor

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

Temas

- Clasificaciones de Log4j2 para Spark
- Ejemplo de configuración de Log4j2 para Spark

- Log4j2 en trabajos de Spark de ejemplo
- Consideraciones de Log4j2 para Spark

Clasificaciones de Log4j2 para Spark

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

`spark-driver-log4j2`

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

`spark-executor-log4j2`

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

Ejemplo de configuración de Log4j2 para Spark

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

Para configurar las clasificaciones de Log4j a nivel de aplicación, en lugar de al enviar el trabajo, consulte Configuración predeterminada de aplicación para EMR sin servidor.

```
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 trabajos de Spark de ejemplo

Los siguientes ejemplos de código muestran cómo crear una aplicación de Spark mientras se inicializa una configuración de 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 ejecute un trabajo de Spark, utilice 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 ejecute un trabajo de Spark, utilice 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 de 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 `log4j2.properties.template` en GitHub.

Monitorización de EMR sin servidor

En esta sección, se describen las formas en las que puede monitorizar sus aplicaciones y trabajos de Amazon EMR sin servidor.

Temas

Monitorización de trabajos y aplicaciones de EMR sin servidor

Con las métricas de Amazon CloudWatch para EMR sin servidor, puede recibir métricas de CloudWatch cada minuto y acceder a los paneles de CloudWatch para acceder en casi tiempo real a las operaciones y el rendimiento de sus aplicaciones de EMR sin servidor.

EMR sin servidor envía métricas a CloudWatch cada minuto. EMR sin servidor emite estas métricas a nivel de aplicación, así como a nivel de trabajo, tipo de trabajador y tipo de asignación de capacidad.

Para empezar, use la plantilla de panel de CloudWatch para EMR sin servidor que se proporciona en el repositorio GitHub de EMR sin servidor y despliéguela.

> **Note**
>
> Las cargas de trabajo interactivas EMR sin servidor solo tienen habilitada la monitorización a nivel de aplicación y tienen una nueva dimensión de tipo de trabajador `Spark_Kernel`. Para monitorizar y depurar sus cargas de trabajo interactivas, acceda a los registros y la IU de Apache Spark desde su espacio de trabajo de EMR Studio.

Supervisión de métricas

> **Important**
>
> Estamos reestructurando nuestra visualización de métricas para agregar `ApplicationName` y `JobName` como dimensiones. En la versión 7.10 y las versiones posteriores, las métricas antiguas ya no se actualizarán. En las versiones de EMR anteriores a la versión 7.10, las métricas antiguas seguirán disponibles.

Dimensiones actuales

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

Dimensiones de las métricas de EMR sin servidor

Dimensión	Descripción	
`ApplicationId`	Filtra todas las métricas de una aplicación de EMR sin servidor mediante el ID de la aplicación.	
`ApplicationName`	Filtra todas las métricas de una aplicación de EMR sin servidor mediante el nombre. Si no se proporcio na el nombre o si contiene caracteres que no sean ASCII, se publica como [Sin especific ar].	
`JobId`	Filtra todas las métricas de una ejecución del trabajo de EMR sin servidor mediante el ID.	
`JobName`	Filtra todas las métricas de una ejecución de trabajo de EMR sin servidor mediante el nombre. Si no se proporcio na el nombre o si contiene caracteres que no sean ASCII, se publica como [Sin especific ar].	
`WorkerType`	Filtra todas las métricas de un tipo de trabajador determina do. Por ejemplo, puede 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 para `PreInitCapacity` por la capacidad preinicializada y `OnDemandCapacity` por todo lo demás.	

Monitorización a nivel de aplicación

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

Métricas de aplicaciones de EMR sin servidor

Métrica	Descripción	Unidad	Dimensión
`MaxCPUAllowed`	El máximo de CPU permitida para la aplicación.	vCPU	`ApplicationId` , `ApplicationName`
`MaxMemoryAllowed`	El máximo de memoria permitida en GB para la aplicación.	Gigabytes (GB)	`ApplicationId` , `ApplicationName`
`MaxStorageAllowed`	El máximo de almacenamiento permitido en GB para la aplicación.	Gigabytes (GB)	`ApplicationId` , `ApplicationName`
`CPUAllocated`	El número total de vCPU asignadas.	vCPU	`ApplicationId` , `ApplicationName` ,

Métrica	Descripción	Unidad	Dimensión
			`WorkerTyp e` , `CapacityA llocationType`
`IdleWorkerCount`	El número total de trabajadores inactivos .	Recuento	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`MemoryAllocated`	La memoria total en GB asignada.	Gigabytes (GB)	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`PendingCr eationWor kerCount`	El número total de trabajadores pendientes de creación.	Recuento	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`RunningWo rkerCount`	El número total de trabajadores que utiliza la aplicación.	Recuento	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`
`StorageAl located`	El almacenamiento total en disco en GB asignado.	Gigabytes (GB)	`ApplicationId` , `ApplicationName` , `WorkerTyp e` , `CapacityA llocationType`

Métrica	Descripción	Unidad	Dimensión
`TotalWorkerCount`	El número total de trabajadores disponibles.	Recuento	`ApplicationId` , `ApplicationName` , `WorkerType` , `CapacityAllocationType`

Monitorización a nivel de trabajo

Amazon EMR sin servidor envía las siguientes métricas a nivel de trabajo a Amazon CloudWatch cada minuto. Puede acceder a los valores de las métricas de las ejecuciones de trabajos agregados por estado de ejecución de trabajos. La unidad de cada una de las métricas es el recuento.

Métricas a nivel de trabajo de EMR sin servidor

Métrica	Descripción	Dimensión
`SubmittedJobs`	El número de trabajos en un estado de Enviado.	`ApplicationId` , `ApplicationName`
`PendingJobs`	El número de trabajos en un estado de Pendiente.	`ApplicationId` , `ApplicationName`
`ScheduledJobs`	El número de trabajos en un estado de Programado.	`ApplicationId` , `ApplicationName`
`RunningJobs`	El número de trabajos en un estado de En ejecución.	`ApplicationId` , `ApplicationName`
`SuccessJobs`	El número de trabajos en un estado de Correcto.	`ApplicationId` , `ApplicationName`
`FailedJobs`	El número de trabajos en un estado de Error.	`ApplicationId` , `ApplicationName`
`CancellingJobs`	El número de trabajos en un estado de Cancelado.	`ApplicationId` , `ApplicationName`

Métrica	Descripción	Dimensión
`CancelledJobs`	El número de trabajos en un estado de Cancelado.	`ApplicationId` , `ApplicationName`

Puede monitorizar las métricas específicas del motor para los trabajos de EMR sin servidor en ejecución y completados con IU de aplicaciones específicas del motor. Cuando se accede a la IU de un trabajo en ejecución, se muestra la IU de la aplicación en directo con actualizaciones en tiempo real. Cuando se accede a la IU de un trabajo completado, se muestra la IU persistente de la aplicación.

Trabajos en ejecución

Para sus trabajos de EMR sin servidor en ejecución, acceda a una interfaz en tiempo real que proporcione métricas específicas del motor. Puede utilizar la IU de Apache Spark o la IU de Hive Tez para monitorizar y depurar sus trabajos. Para acceder a estas IU, utilice la consola de EMR Studio o solicite un punto de conexión URL seguro con la AWS Command Line Interface.

Trabajos completados

Para sus trabajos de EMR sin servidor completados, utilice el servidor de historial de Spark o la IU persistente de Hive Tez para acceder a los detalles de los trabajos, las etapas, las tareas y las métricas de las ejecuciones de trabajos de Spark o Hive. Para acceder a estas IU, utilice la consola de EMR Studio o solicite un punto de conexión URL seguro con la AWS Command Line Interface.

Monitorización a nivel de trabajador laboral

Amazon EMR sin servidor envía a Amazon CloudWatch las siguientes métricas a nivel de trabajador laboral que están disponibles en el espacio de nombres `AWS/EMRServerless` y el grupo de métricas `Job Worker Metrics`. EMR sin servidor recopila puntos de datos de trabajadores individuales durante la ejecución de los trabajos a nivel de trabajo, tipo de trabajador y tipo de asignación de capacidad. Se puede utilizar `ApplicationId` como una dimensión para monitorizar varios trabajos que pertenecen a la misma aplicación.

Métricas a nivel de trabajador laboral de EMR sin servidor

Métrica	Descripción	Unidad	Dimensión
`WorkerCpuAllocated`	El número total de núcleos de vCPU asignados a los trabajadores en una ejecución de trabajo.	vCPU	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`
`WorkerCpuUsed`	El número total de núcleos de vCPU utilizados por los trabajadores en una ejecución de trabajo.	vCPU	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`
`WorkerMemoryAllocated`	La memoria total en GB asignada a los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`

Métrica	Descripción	Unidad	Dimensión	
`WorkerMemoryUsed`	La memoria total en GB utilizada por los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`	
`WorkerEphemeralStorageAllocated`	El número de bytes de almacenamiento efímero asignados a los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`	
`WorkerEphemeralStorageUsed`	El número de bytes de almacenamiento efímero utilizados por los trabajadores en una ejecución de trabajo.	Gigabytes (GB)	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`	

Métrica	Descripción	Unidad	Dimensión	
`WorkerStorageReadBytes`	El número de bytes leídos del almacenamiento por los trabajadores en una ejecución de trabajo.	Bytes	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`	
`WorkerStorageWriteBytes`	El número de bytes escritos en almacenamiento por los trabajadores en una ejecución de trabajo.	Bytes	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, y `CapacityAllocationType`	

En los siguientes pasos, se describe cómo acceder a los distintos tipos de métricas.

Console

Para acceder a la IU de la aplicación con la consola

1. Navegue hasta la aplicación EMR sin servidor en EMR Studio siguiendo las instrucciones de Introducción a la consola.
2. Para acceder a las UI y los registros de aplicaciones específicas del motor para un trabajo en ejecución, realice lo siguiente:

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

b. Seleccione el trabajo en la página de Detalles de la aplicación o vaya a la página de Detalles del trabajo correspondiente a su trabajo.

c. En el menú desplegable Mostrar IU, seleccione o la IU de Spark o la IU de Hive Tez para ir a la aplicación de IU correspondiente a su tipo de trabajo.

d. Para acceder a los registros del motor de Spark, vaya a la pestaña Ejecutores de la IU de Spark y seleccione el enlace Registros del controlador. Para acceder a los registros del motor de Hive, seleccione el enlace Registros del DAG correspondiente en la IU de Hive Tez.

3. Para acceder a las IU y los registros de aplicaciones específicas del motor para un trabajo completado, realice lo siguiente:

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

b. Seleccione el trabajo en la página Detalles de la aplicación de la aplicación o vaya a la página de Detalles del trabajo de su trabajo.

c. En el menú desplegable Mostrar IU, seleccione o el Servidor del historial de Spark o la IU persistente de Hive Tez para ir a la IU de la aplicación correspondiente a su tipo de trabajo.

d. Para acceder a los registros del motor de Spark, vaya a la pestaña Ejecutores de la IU de Spark y seleccione el enlace Registros del controlador. Para acceder a los registros del motor de Hive, seleccione el enlace Registros del DAG correspondiente en la IU de Hive Tez.

AWS CLI

Para acceder a la IU de la aplicación con la AWS CLI

- Para generar una URL que pueda usar para acceder a la IU de su aplicación, tanto para los trabajos en ejecución como para los completados, llame a la API `GetDashboardForJobRun`.

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

La URL que genere es válida durante una hora.

Monitorice las métricas de Spark con Amazon Managed Service para Prometheus

Con las versiones 7.1.0 y posteriores de Amazon EMR, puede integrar EMR sin servidor con Amazon Managed Service para Prometheus para recopilar métricas de Apache Spark para trabajos y aplicaciones de EMR sin servidor. Esta integración está disponible cuando envía un trabajo o crea una aplicación mediante la consola AWS, la API de EMR sin servidor, o la AWS CLI.

Requisitos previos

Antes de enviar sus métricas de Spark a Amazon Managed Service para Prometheus, debe cumplir los siguientes requisitos previos.

- Crear un espacio de trabajo de Amazon Managed Service para Prometheus. Este espacio de trabajo sirve como punto de conexión de ingestión. Anote la URL que se muestra para el punto de conexión: URL de escritura remota. Deberá especificar la URL al crear la aplicación EMR sin servidor.
- Para conceder acceso a sus trabajos a Amazon Managed Service para Prometheus con fines de monitorización, añada la siguiente política a su rol 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 la consola AWS para crear una aplicación que esté integrada con Amazon Managed Service para Prometheus

1. Consulte Introducción a Amazon EMR sin servidor 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, seleccione Entregar las métricas del motor a Amazon Managed Service para Prometheus y, a continuación, especifique su URL de escritura remota.

4. Especifique cualquier otro ajuste de configuración que desee y, a continuación, seleccione Crear e iniciar la aplicación.

Utilice la AWS CLI o la API de EMR sin servidor

También puede utilizar la AWS CLI o la API de EMR sin servidor para integrar su aplicación EMR sin servidor con Amazon Managed Service para Prometheus cuando ejecute los comandos del `create-application` o del `start-job-run`.

create-application

```
aws emr-serverless create-application \
--release-label emr-7.1.0 \
--type "SPARK" \
--monitoring-configuration '{
    "prometheusMonitoringConfiguration": {
        "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
    }
}'
```

start-job-run

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
        "entryPointArguments": ["10000"],
        "sparkSubmitParameters": "--conf spark.dynamicAllocation.maxExecutors=10"
    }
}' \
--configuration-overrides '{
     "monitoringConfiguration": {
        "prometheusMonitoringConfiguration": {
            "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
        }
    }
}'
```

Incluir `prometheusMonitoringConfiguration` en su comando indica que EMR sin servidor debe ejecutar el trabajo de Spark con un agente que recopile las métricas de Spark y las escriba en su punto de conexión `remoteWriteUrl` para Amazon Managed Service para Prometheus. A continuación, puede utilizar las métricas de Spark en Amazon Managed Service para Prometheus para la visualización, las alertas y el análisis.

Propiedades de configuración avanzada

EMR sin servidor 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 para Prometheus. De forma predeterminada, EMR sin servidor establece los valores predeterminados en Spark y analiza las métricas de los controladores y ejecutores cuando envía un trabajo mediante `PrometheusMonitoringConfiguration`.

En la siguiente tabla, se describen todas las propiedades que puede configurar cuando envíe un trabajo de Spark que envíe métricas a Amazon Managed Service para Prometheus.

Propiedad de Spark	Valor predeterminado	Descripción
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	La clase que Spark utiliza para enviar métricas a Amazon Managed Service para Prometheus. Para anular el comportamiento predeterminado, especifique su propia clase personalizada.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source.jvmSource	La clase que Spark utiliza para recopilar y enviar métricas cruciales desde la máquina virtual Java subyacente. Para dejar de recopilar métricas de JVM, deshabilite 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	La URL diferente que Amazon Managed Service para Prometheus utiliza para recopilar las métricas del controlador. Para anular el comportamiento predeterminado, especifique su propia ruta. Para dejar de recopilar métricas del controlador, deshabilite esta propiedad configurándola en una cadena vacía, como "".
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/executor/prometheus	La URL diferente que Amazon Managed Service para Prometheus utiliza para recopilar las métricas del ejecutor. Para anular el comportamiento predeterminado, especifique su propia ruta. Para dejar de recopilar métricas del ejecutor, deshabilite esta propiedad configurándola en una cadena vacía, como "".

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

Consideraciones y limitaciones

Cuando utilice Amazon Managed Service para Prometheus para recopilar métricas de EMR sin servidor, tenga en cuenta las siguientes consideraciones y limitaciones.

- El soporte técnico para usar Amazon Managed Service para Prometheus con EMR sin servidor solo está disponible en aquellas Regiones de AWS donde Amazon Managed Service para Prometheus esté disponible de forma general.
- Ejecutar el agente para recopilar las métricas de Spark en Amazon Managed Service para Prometheus requiere más recursos por parte de los trabajadores. Si elige un tamaño de trabajador más pequeño, como un trabajador de vCPU, el tiempo de ejecución de su trabajo podría aumentar.
- La compatibilidad con Amazon Managed Service para Prometheus con EMR sin servidor solo está disponible para las versiones 7.1.0 y posteriores de Amazon EMR.
- Amazon Managed Service para Prometheus debe implementarse en la misma cuenta en la que ejecuta EMR sin servidor para poder recopilar métricas.

Métricas de uso de EMR sin servidor

Puede utilizar las métricas de uso de Amazon CloudWatch para proporcionar visibilidad sobre los recursos que usa su cuenta. Utilice estas métricas para visualizar el uso del servicio en paneles y gráficos de CloudWatch.

Las métricas de uso de EMR sin servidor se corresponden con las 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 alarmas de Amazon CloudWatch en la Guía del usuario de Service Quotasi.

Para obtener más información acerca de las Service Quotas de EMR sin servidor, consulte Cuotas y puntos de conexión para EMR Serverless.

Métricas de uso de Service Quotas para EMR sin servidor

EMR sin servidor publica las siguientes métricas de uso de Service Quotas en el espacio de nombres `AWS/Usage`.

Métrica	Descripción
`ResourceCount`	El número total de los recursos especificados que se ejecutan en su cuenta. Los recursos se definen por las dimensiones que se asocian con la métrica.

Dimensiones para las métricas de uso de Service Quotas de EMR sin servidor

Puede utilizar las siguientes dimensiones para ajustar las métricas de uso que publica EMR sin servidor.

Dimensión	Valor	Descripción
`Service`	EMR sin servidor	El nombre del Servicio de AWS que contiene el recurso.
`Type`	Recurso	El tipo de entidad que EMR sin servidor comunica.
`Resource`	vCPU	El tipo de recurso que EMR sin servidor sigue.
`Class`	Ninguno	La clase de recurso que EMR sin servidor sigue.

Automatización de EMR sin servidor con Amazon EventBridge

Puede utilizar Amazon EventBridge para automatizar sus Servicios de AWS y responder automáticamente a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios de recursos. EventBridge proporciona una secuencia de eventos de sistema casi en tiempo real que describen cambios en sus recursos de AWS. 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 una función de AWS Lambda
- Transmitir un evento a Amazon Kinesis Data Streams
- Activar una máquina de estado de AWS Step Functions
- Notificación sobre un tema de Amazon SNS o una cola de Amazon SQS

Por ejemplo, si utiliza EventBridge con EMR sin servidor, puede activar una función AWS Lambda cuando un trabajo de ETL se realice correctamente o notificar a un tema de Amazon SNS cuando un trabajo de ETL dé error.

EMR sin servidor emite cuatro 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 acerca lo que indica la aplicación, consulte Estados de la aplicación.
- Eventos de cambio de estado de ejecución de un trabajo: eventos que emiten todos los cambios de estado de una ejecución de trabajo. Para obtener más información sobre esto, consulte Estados de ejecuciones de trabajos.
- Eventos de reintento de ejecución de trabajos: eventos que emiten cada reintento de ejecución de un trabajo desde Amazon EMR sin servidor, versión 7.1.0 y posteriores.
- Eventos de actualización de uso de recursos de trabajo: eventos que emiten actualizaciones de uso de recursos para un trabajo que se ejecutan en intervalos cercanos a 30 minutos.

Ejemplos de eventos de EMR sin servidor EventBridge

Los eventos notificados por EMR sin servidor tienen un valor `aws.emr-serverless` asignado a la `source`, como en los ejemplos siguientes.

Evento de cambio de estado de la aplicación

El siguiente evento de ejemplo muestra una aplicación en el estado `CREATING`.

```
{
    "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 de ejecución de trabajo

El siguiente evento de ejemplo muestra una ejecución de trabajo que se mueve del estado `SCHEDULED` al estado `RUNNING`.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "state": "RUNNING",
        "previousState": "SCHEDULED",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z"
    }
}
```

Evento de reintento de ejecución de trabajo

A continuación, se muestra un ejemplo de un evento de reintento de ejecución de 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"
    }
}
```

Actualización sobre la utilización de los recursos del trabajo

El siguiente evento de ejemplo muestra la actualización final de utilización de recursos de un trabajo que pasó a un estado terminal después de ejecutarse.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Resource Utilization Update",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [
```

```
        "arn:aws:emr-serverless:us-east-1:123456789012:/applications/00f1982r1uukb925/
jobruns/00f1cbn5g4bb0c01"
    ],
    "detail": {
        "applicationId": "00f1982r1uukb925",
        "jobRunId": "00f1cbn5g4bb0c01",
        "attempt": 1,
        "mode": "BATCH",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        "startedAt": "2022-05-31T21:07:26.123Z",
        "calculatedFrom": "2022-05-31T21:07:42.299487Z",
        "calculatedTo": "2022-05-31T21:07:30.325900Z",
        "resourceUtilizationFinal": true,
        "resourceUtilizationForInterval": {
            "vCPUHour": 0.023,
            "memoryGBHour": 0.114,
            "storageGBHour": 0.228
        },
        "billedResourceUtilizationForInterval": {
            "vCPUHour": 0.067,
            "memoryGBHour": 0.333,
            "storageGBHour": 0
        },
        "totalResourceUtilization": {
            "vCPUHour": 0.023,
            "memoryGBHour": 0.114,
            "storageGBHour": 0.228
        },
        "totalBilledResourceUtilization": {
            "vCPUHour": 0.067,
            "memoryGBHour": 0.333,
            "storageGBHour": 0
        }
    }
}
```

El campo startedAt solo estará presente en el caso de que el trabajo haya pasado a un estado de ejecución.

Etiquetado de recursos

Asigne sus propios metadatos a cada recurso mediante etiquetas para ayudarlo a administrar sus recursos de EMR sin servidor. Esta sección proporciona información general sobre las funciones de las etiquetas y le muestra cómo crear etiquetas.

Temas

¿Qué es una etiqueta?

Una etiqueta es una marca que se asigna a un recurso de AWS. Cada etiqueta consta de una clave y un valor, ambos definidos por el usuario. Las etiquetas permiten clasificar los recursos de AWS de diversas maneras, por ejemplo, según su finalidad, propietario o entorno. Cuando tenga muchos recursos del mismo tipo, identifique rápidamente un recurso específico en función de las etiquetas que le haya asignado. Por ejemplo, defina un conjunto de etiquetas para sus aplicaciones de Amazon EMR sin servidor que lo ayuden a realizar un seguimiento del propietario y del nivel de pila de cada aplicación. Le sugerimos 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 agregar una etiqueta a un recurso, modifique el valor de la etiqueta o elimine la etiqueta del recurso en cualquier momento. Las etiquetas no tienen ningún significado semántico para Amazon EMR sin servidor, por lo que 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 utiliza IAM, puede controlar qué usuarios de su cuenta de AWS tienen permiso para administrar 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 trabajo nuevas o existentes. Si utiliza la API de Amazon EMR sin servidor, la AWS CLI o un SDK de AWS, puede aplicar etiquetas a los recursos nuevos mediante el parámetro `tags` en la acción de la API pertinente. Puede aplicar etiquetas a recursos existentes a través de la acción de la API `TagResource`.

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 recursos en el momento de su creación, no es necesario ejecutar scripts de etiquetado personalizados después de la creación del recurso.

En la siguiente tabla se describen los recursos de Amazon EMR sin servidor que admiten etiquetas.

Recursos etiquetables de

Recurso	Admite etiquetas	Admite la propagación de etiquetas	Admite el etiquetado durante la creación (API de Amazon EMR sin servidor, la AWS CLI y el SDK de AWS)	API para creación (se pueden agregar etiquetas 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 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 y espacios representables en UTF-8, además de 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 `AWS:` ni ninguna combinación de mayúsculas o minúsculas del mismo como prefijo para claves o valores. Estos están reservados solo para la utilización de AWS.

Trabajo con etiquetas mediante la AWS CLI y la API de Amazon EMR servidor

Utilice los siguientes comandos de la AWS CLI o las operaciones de la API de Amazon EMR sin servidor para agregar, actualizar, enumerar y eliminar las etiquetas de sus recursos.

Comandos de CLI y operaciones de API para etiquetas

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`

Los siguientes ejemplos muestran cómo agregar o quitar etiquetas a los recursos mediante la AWS CLI.

Etiquetar una aplicación existente

El siguiente comando etiqueta una aplicación existente.

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

Desetiquetar 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
```

Enumerar 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 EMR sin servidor

Esta sección describe casos de uso comunes para cuando trabaje con las aplicaciones de EMR sin servidor. Se tratan distintas herramientas, incluidas Hudi e Iceberg, para trabajar con grandes conjuntos de datos, así como el uso de Python y bibliotecas de Python para enviar trabajos a Spark.

Temas

Uso de Java 17 con Amazon EMR sin servidor

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

`JAVA_HOME`

Para anular la configuración de JVM para EMR sin servidor 6.11.0 y versiones posteriores, incluya la configuración `JAVA_HOME` en sus clasificaciones de entorno `spark.emr-serverless.driverEnv` y `spark.executorEnv`.

x86_64

Establezca las propiedades necesarias para especificar Java 17 como configuración `JAVA_HOME` 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 configuración `JAVA_HOME` 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, puede especificar Java 17 en la clasificación `spark-defaults` para anular la configuración de JVM para EMR sin servidor en la versión 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 clasificación `spark-defaults`:

```
{
```

```
"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 EMR sin servidor

En esta sección se describe el uso de Apache Hudi con aplicaciones EMR sin servidor. Hudi es un marco de administración de datos que simplifica el procesamiento de datos.

Para usar Apache Hudi con aplicaciones EMR sin servidor

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

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

2. Para sincronizar una tabla de Hudi con el catálogo configurado, designe el catálogo de datos de Glue AWS como metaalmacén o configure un metaalmacén externo. EMR sin servidor admite `hms` como modo de sincronización de las tablas Hive para las cargas de trabajo de Hudi. EMR sin servidor activa esta propiedad de forma predeterminada. Para obtener más información acerca de cómo configurar su metaalmacén, consulte Configuración de metaalmacenes para EMR sin servidor.

   > **Important**
   >
   > EMR sin servidor no admite `HIVEQL` ni `JDBC` como opciones de modo de sincronización para que las tablas Hive gestionen las cargas de trabajo de Hudi. Para obtener más información, consulte Sync modes.

Cuando utilice el catálogo de datos de AWS Glue como metaalmacén, especifique 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 Amazon EMR, consulte Hudi release history.

Uso de Apache Iceberg con EMR sin servidor

En esta sección se describe cómo utilizar Apache Iceberg con aplicaciones EMR sin servidor. Apache Iceberg es un formato de tabla que ayuda a trabajar con grandes conjuntos de datos en lagos de datos.

Para usar Apache Iceberg con aplicaciones EMR sin servidor

1. Establezca 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 el Catálogo de datos de Glue AWS como metaalmacén o configure un metaalmacén externo. Para obtener más información acerca de cómo configurar su metaalmacén, consulte Configuración de metaalmacenes para EMR sin servidor.

   Configure las propiedades del metaalmacén que desee utilizar para Iceberg. Por ejemplo, si desea utilizar el Catálogo de datos de Glue AWS, establezca las siguientes propiedades en la configuración de la aplicación.

   ```
   spark.sql.catalog.dev.warehouse=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
   spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
   spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog
   spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
   spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
   ```

Cuando utilice el catálogo de datos de AWS Glue como metaalmacén, puede especificar las siguientes propiedades de configuración para tu 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://amzn-s3-demo-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 Amazon EMR, consulte Iceberg release history.

Uso de bibliotecas de Python con EMR sin servidor

Cuando ejecute trabajos de PySpark en aplicaciones Amazon EMR sin servidor, empaquete varias bibliotecas de Python como dependencias. Para ello, use características nativas de Python, cree un entorno virtual o configure directamente sus trabajos de PySpark para que usen bibliotecas de Python. Esta página abarca cada enfoque.

Uso de funciones nativas de Python

Si establece la siguiente configuración, use 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://amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
 file>
```

Para obtener más información sobre cómo usar los entornos virtuales de Python para los trabajos de PySpark, consulte Using PySpark Native Features.

Al usar EMR Notebook, puede hacer que la dependencia de Python esté disponible en su cuaderno mediante la ejecución del siguiente código:

```
    %%configure -f
```

```
{
   "conf": {
   "spark.submit.pyFiles":"s3:///amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
file>
                  }
}
```

Creación de un entorno virtual de Python

Para empaquetar varias bibliotecas de Python para un trabajo de PySpark, cree un entorno virtual de Python aislado.

1. Para crear el entorno virtual de Python, ejecute los siguientes comandos. El ejemplo que se muestra instala los paquetes `scipy` y `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 sin servidor, es decir, Python 3.7.10 para Amazon EMR versión 6.6.0. Puede encontrar un ejemplo de Dockerfile en el repositorio GitHub de Ejemplos de EMR sin servidor.

   ```
   # initialize a python virtual environment
   python3 -m venv pyspark_venvsource
   source pyspark_venvsource/bin/activate

   # optionally, ensure pip is up-to-date
   pip3 install --upgrade pip

   # install the python packages
   pip3 install scipy
   pip3 install matplotlib

   # package the virtual environment into an archive
   pip3 install venv-pack
   venv-pack -f -o pyspark_venv.tar.gz

   # copy the archive to an S3 location
   aws s3 cp pyspark_venv.tar.gz s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
   ```

```
# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Envíe el trabajo de Spark con sus propiedades configuradas para usar el entorno virtual de Python.

```
--conf spark.archives=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

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 usar los entornos virtuales de Python para los trabajos de PySpark, consulte Using Virtualenv. Para ver más ejemplos de cómo enviar trabajos de Spark, consulte Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor.

Configuración de trabajos de PySpark para usar bibliotecas de Python

Con las versiones 6.12.0 y posteriores de Amazon EMR, puede configurar directamente los trabajos de PySpark sin servidor de EMR para utilizar bibliotecas Python populares de ciencia de datos, como pandas, NumPy y PyArrow, sin necesidad de realizar ninguna configuración adicional.

En los siguientes ejemplos, se muestra cómo empaquetar cada biblioteca de Python para un trabajo de PySpark.

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 usar NumPy, ejecute el siguiente comando:

```
import numpy
```

pandas

pandas es una biblioteca de Python construida sobre NumPy. La biblioteca pandas proporciona a los científicos de datos estructuras de datos DataFrame y herramientas de análisis de datos. Para usar pandas, ejecute 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 usar PyArrow, ejecute el siguiente comando:

```
import pyarrow
```

Uso de diferentes versiones de Python con EMR sin servidor

Además del caso de uso en Uso de bibliotecas de Python con EMR sin servidor, 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 sin servidor. 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 Python

1. Cree el entorno virtual con los comandos del ejemplo siguiente. 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 utiliza Amazon EMR versión 7.0.0 y versiones posteriores, ejecute sus comandos en un entorno Amazon Linux 2023 similar al que utiliza para sus aplicaciones EMR sin servidor.
> Si utiliza la versión 6.15.0 o una versión anterior, ejecute 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. Configure sus propiedades para usar el entorno virtual de Python y envíe 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://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Para obtener más información sobre cómo usar los entornos virtuales de Python para los trabajos de PySpark, consulte Using Virtualenv. Para ver más ejemplos de cómo enviar trabajos de Spark, consulte Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor.

Uso de OSS de Delta Lake con EMR sin servidor

Versiones 6.9.0 y posteriores de Amazon EMR

> **Note**
>
> Las versiones 7.0.0 y posteriores de Amazon EMR utilizan Delta Lake 3.0.0, que cambia el nombre del archivo `delta-core.jar` a `delta-spark.jar`. Si utiliza Amazon EMR 7.0.0 o posterior, asegúrese de especificar `delta-spark.jar` en la configuración.

Las versiones Amazon EMR 6.9.0 y versiones posteriores incluyen Delta Lake, por lo que ya no tiene que empaquetar Delta Lake usted mismo ni proporcionar la marca `--packages` con sus trabajos de 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 `delta_sample.py` local para probar la creación y lectura de una tabla Delta.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://amzn-s3-demo-bucket/delta-lake/output/%s/" % str(uuid.uuid4())
    spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

    ## creates a Delta table and outputs to target S3 bucket
    spark.range(5).write.format("delta").save(url)
```

```
    ## reads a Delta table and outputs to target S3 bucket
    spark.read.format("delta").load(url).show
```

3. Cargue el archivo `delta_sample.py`, suba el archivo AWS CLI a su bucket de Amazon S3. A continuación, utilice el comando `start-job-run` para enviar un trabajo a una aplicación EMR sin servidor existente.

```
aws s3 cp delta_sample.py s3://amzn-s3-demo-bucket/code/

    aws emr-serverless start-job-run \
        --application-id application-id \
        --execution-role-arn job-role-arn \
        --name emr-delta \
        --job-driver '{
            "sparkSubmit": {
                "entryPoint": "s3://amzn-s3-demo-bucket/code/delta_sample.py",
                "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
            }
        }'
```

Para usar bibliotecas de Python con Delta Lake, agregue la biblioteca `delta-core` empaquetándola como una dependencia o usándola como una imagen personalizada.

Alternativamente, puede usar `SparkContext.addPyFile` para agregar las bibliotecas de Python desde el archivo JAR `delta-core`:

```
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])
```

Versiones 6.8.0 y posteriores de Amazon EMR

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 sin servidor, navegue hasta Delta GitHub y siga las instrucciones.
2. Cargar las bibliotecas de Delta Lake en el bucket de Amazon S3 de su Cuenta de AWS.
3. Cuando envíe trabajos EMR sin servidor en la configuración de la aplicación, incluya los archivos JAR de Delta Lake que se encuentran ahora en el bucket.

```
--conf spark.jars=s3://amzn-s3-demo-bucket/jars/delta-core_2.12-1.1.0.jar
```

4. Para asegurarte de que puedes leer y escribir desde una tabla Delta, ejecuta un ejemplo de prueba con PySpark.

```
from pyspark import SparkConf, SparkContext
    from pyspark.sql import HiveContext, SparkSession

    import uuid

    conf = SparkConf()
    sc = SparkContext(conf=conf)
    sqlContext = HiveContext(sc)

    url = "s3://amzn-s3-demo-bucket/delta-lake/output/1.0.1/%s/" %
 str(uuid.uuid4())

    ## creates a Delta table and outputs to target S3 bucket
    session.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    session.read.format("delta").load(url).show
```

Envío de trabajos de 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.

Puede utilizar `EmrServerlessCreateApplicationOperator` para crear una aplicación Spark o Hive. También puede utilizar `EmrServerlessStartJobOperator` para iniciar uno o más trabajos con su nueva aplicación.

Para usar el operador con Amazon Managed Workflows for Apache Airflow (MWAA) con Airflow 2.2.2, añada la siguiente línea al archivo `requirements.txt` y actualice su entorno de MWAA para usar el nuevo archivo.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Tenga en cuenta que la compatibilidad con EMR sin servidor 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 MWAA.

El siguiente ejemplo abreviado muestra cómo crear una aplicación, ejecutar varios trabajos de Spark y, a continuación, detener la aplicación. Un ejemplo completo está disponible en el repositorio GitHub de Ejemplos de EMR sin servidor. Para obtener información adicional de la configuración `sparkSubmit`, consulte Uso de configuraciones de Spark al ejecutar trabajos de EMR sin servidor.

```
from datetime import datetime

from airflow import DAG
from airflow.providers.amazon.aws.operators.emr import (
    EmrServerlessCreateApplicationOperator,
    EmrServerlessStartJobOperator,
    EmrServerlessDeleteApplicationOperator,
)

# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "amzn-s3-demo-bucket"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://amzn-s3-demo-bucket/logs/"}
    },
}

with DAG(
    dag_id="example_endtoend_emr_serverless_job",
    schedule_interval=None,
    start_date=datetime(2021, 1, 1),
    tags=["example"],
    catchup=False,
) as dag:
```

```
    create_app = EmrServerlessCreateApplicationOperator(
        task_id="create_spark_app",
        job_type="SPARK",
        release_label="emr-6.7.0",
        config={"name": "airflow-test"},
    )

    application_id = create_app.output

    job1 = EmrServerlessStartJobOperator(
        task_id="start_job_1",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    job2 = EmrServerlessStartJobOperator(
        task_id="start_job_2",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
                "entryPointArguments": ["1000"]
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    delete_app = EmrServerlessDeleteApplicationOperator(
        task_id="delete_app",
        application_id=application_id,
        trigger_rule="all_done",
    )

    (create_app >> [job1, job2] >> delete_app)
```

Uso de funciones definidas por el usuario de Hive con EMR sin servidor

Las funciones definidas por el usuario (UDF) le permiten crear funciones personalizadas para procesar registros o grupos de registros. En este tutorial, utilizará un ejemplo de UDF con una aplicación Amazon EMR sin servidor preexistente para ejecutar un trabajo que genere un resultado de consulta. Para obtener información acerca de cómo configurar una aplicación, consulte Introducción a Amazon EMR sin servidor.

Para usar una UDF con EMR sin servidor

1. Vaya a GitHub para ver un ejemplo de UDF. Clone el repositorio y cambie a la rama de git que quiera usar. Actualice el `maven-compiler-plugin` en el archivo `pom.xml` del repositorio para tener una fuente. Actualice también la configuración de la versión java de destino a `1.8`. Ejecute `mvn package -DskipTests` para crear el archivo JAR que contiene las UDF de muestra.
2. Tras crear el archivo JAR, cárguelo a su bucket de S3 con el siguiente comando.

```
aws s3 cp brickhouse-0.8.2-JS.jar s3://amzn-s3-demo-bucket/jars/
```

3. Cree un archivo de ejemplo para usar una de las funciones UDF de ejemplo. Guarde esta consulta como `udf_example.q` y cárguela en su bucket de S3.

```
add jar s3://amzn-s3-demo-bucket/jars/brickhouse-0.8.2-JS.jar;
CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))));
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))))["key1"][2];
```

4. Envíe 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://amzn-s3-demo-bucket/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/
emr-serverless-hive/warehouse"
```

```
    }
}' --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.driver.cores": "2",
            "hive.driver.memory": "6G"
        }
    }],
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/logs/"
        }
    }
}'
```

5. Utilice el comando `get-job-run` para comprobar el estado de su trabajo. Espere 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://amzn-s3-demo-bucket/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

El archivo `stdout.gz` se parece al siguiente.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

Uso de imágenes personalizadas con EMR sin servidor

Temas

- Use una versión personalizada de Python
- Use una versión Java personalizada
- Cree una imagen de ciencia de datos
- Procesamiento de datos geoespaciales con Apache Sedona

- Información sobre licencias para el uso de imágenes personalizadas

Use 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 runs the image as hadoop
USER hadoop:hadoop
```

Antes de enviar el trabajo de Spark, configure sus 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
```

Use una versión Java personalizada

En el siguiente ejemplo, se muestra cómo crear una imagen personalizada para usar Java 11 en los trabajos de Spark.

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

USER root

# install JDK 11
RUN amazon-linux-extras install java-openjdk11
```

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

Antes de enviar el trabajo de Spark, configure 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-
```

Cree 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 runs the image as hadoop
USER hadoop:hadoop
```

Procesamiento de datos geoespaciales con Apache Sedona

En el siguiente ejemplo se 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 runs the image as hadoop
USER hadoop:hadoop
```

Información sobre licencias para el uso de imágenes personalizadas

Puede crear imágenes personalizadas con EMR sin servidor para realizar tareas específicas o utilizar versiones específicas de un paquete de software. La modificación y distribución de imágenes personalizadas pueden estar sujetas a normas y condiciones de licencia. El texto de la licencia aparece en la subsección siguiente.

Licencia que se aplica a las imágenes personalizadas

Derechos de autor de Amazon.com y sus filiales; todos los derechos reservados. Este software es contenido de AWS que se incluye en el Acuerdo con el cliente de AWS y no se puede distribuir sin permiso. Además de los permisos de la Licencia de propiedad intelectual de AWS, el Licenciante de AWS le concede los siguientes permisos adicionales:

Se permite crear, copiar y utilizar derivados del contenido de AWS siempre que se cumplan las siguientes condiciones:

- Usted no modifica el contenido de AWS en sí, y cualquier derivado es estrictamente el resultado de la adición de contenido nuevo por su parte.
- Las reproducciones internas deben conservar el aviso de derechos de autor anterior.
- La distribución externa, en formato fuente o binario, con o sin modificaciones, no está permitida según los términos de esta licencia.

Para obtener más información sobre el uso de imágenes personalizadas, consulte Using custom images with EMR Serverless.

Uso de la integración de Amazon Redshift para Apache Spark en Amazon EMR sin servidor

Con la versión 6.9.0 y posteriores de Amazon EMR, la imagen de cada versión incluye un conector entre Apache Spark y Amazon Redshift. Con este conector, use Spark en Amazon EMR sin servidor para procesar los datos almacenados en Amazon Redshift. La integración se basa en el conector de

código abierto `spark-redshift`. Para Amazon EMR sin servidor en EKS, la integración de Amazon Redshift para Apache Spark se incluye como una integración nativa.

Temas

Lanzamiento de una aplicación de Spark mediante la integración de Amazon Redshift para Apache Spark

Para usar la integración con EMR sin servidor 6.9.0, debe pasar las dependencias de Spark Redshift necesarias con su trabajo de Spark. Utilice `--jars` para incluir bibliotecas relacionadas con el conector de Redshift. Para acceder a otras ubicaciones de archivos compatibles con la opción `--jars`, consulte la sección Advanced Dependency Management de la documentación de Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Las versiones 6.10.0 y posteriores de Amazon EMR no requieren la dependencia `minimal-json.jar` e instalan automáticamente las demás dependencias en cada clúster de forma predeterminada. En los siguientes ejemplos, se demuestra 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 sin servidor con la integración de Amazon Redshift para Apache Spark en EMR sin servidor 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 sin servidor con la integración de Amazon Redshift para Apache Spark en EMR sin servidor 6.9.0, utilice la opción `--jars` que se muestra en el siguiente ejemplo. Tenga en cuenta que las rutas enumeradas con la opción `--jars` son las rutas predeterminadas para 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

Utilice AWS Secrets Manager para recuperar 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 a la API de `GetSecretValue` para obtenerlas:

```

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 de JDBC

Establecer el nombre de usuario y la contraseña dentro de la URL de JDBC

Puede autenticar un trabajo de Spark en un clúster de Amazon Redshift al especificar el nombre y la contraseña de la base de datos de Amazon Redshift en la URL de JDBC.

> **Note**
>
> Si pasa las credenciales de la base de datos en la URL, cualquier persona que tenga acceso a la URL también 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, use el siguiente formato para configurar el nombre de usuario y la contraseña en la URL de JDBC:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Utilizar la autenticación basada en IAM con el rol de ejecución de trabajos de Amazon EMR sin servidor

A partir de la versión 6.9.0 de Amazon EMR sin servidor, la versión 2.1 o posterior del controlador de JDBC de Amazon Redshift se incluye en el entorno. Con el controlador JDBC 2.1 y versiones posteriores, puede especificar la URL de JDBC sin incluir el nombre de usuario y la contraseña sin encriptar.

En su lugar, especifique el esquema `jdbc:redshift:iam://`. Esto ordena al controlador de JDBC que utilice su rol de ejecución de trabajos de EMR sin servidor para obtener las credenciales automáticamente. Para obtener más información, consulte Configurar una conexión JDBC u ODBC para usar credenciales de IAM en la Guía de administración de Amazon Redshift. Un ejemplo de esta URL es:

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Los siguientes permisos son necesarios para su rol de ejecución de trabajos si cumple con las condiciones proporcionadas:

Permiso	Condiciones en las que se requiere un rol de ejecución de trabajos
`redshift:GetClusterCredentials`	Obligatorio para que el controlador de JDBC obtenga las credenciales de Amazon Redshift
`redshift:DescribeCluster`	Obligatorio si especifica el clúster de Amazon Redshift y la Región de AWS en la URL de JDBC en lugar del punto de conexión
`redshift-serverless:GetCredentials`	Obligatorio para que el controlador de JDBC obtenga las credenciales de Amazon Redshift sin servidor
`redshift-serverless:GetWorkgroup`	Obligatorio si utiliza Amazon Redshift sin servidor y especifica la URL en términos de nombre y región del grupo de trabajo

Conexión a Amazon Redshift desde una VPC diferente

Al configurar un clúster de Amazon Redshift aprovisionado o un grupo de trabajo Amazon Redshift sin servidor en una VPC, configure la conectividad de VPC para que su aplicación Amazon EMR sin servidor pueda acceder a los recursos. Para obtener más información sobre cómo configurar la conectividad de VPC en una aplicación EMR sin servidor, consulte Configuración del acceso a la VPC para que las aplicaciones EMR sin servidor se conecten a los datos.

- Si su clúster de Amazon Redshift o grupo de trabajo Amazon Redshift sin servidor aprovisionado es de acceso público, especifique una o más subredes privadas que tengan una puerta de enlace NAT conectada al crear aplicaciones EMR sin servidor.
- 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 punto de enlace de VPC gestionado por Amazon Redshift para su clúster de Amazon Redshift, tal y como se describe en Configuración del acceso a la VPC para que las aplicaciones EMR sin servidor se conecten a los datos. Como

alternativa, puede crear su grupo de trabajo Amazon Redshift Serverless tal y como se describe en Conexión a Amazon Redshift sin servidor, en la Guía de administración de Amazon Redshift. Debe asociar el clúster o el subgrupo a las subredes privadas que especifique al crear la aplicación EMR sin servidor.

> **Note**
>
> Si utiliza la autenticación basada en IAM y las subredes privadas de la aplicación EMR sin servidor no tienen una puerta de enlace NAT adjunta, también debe crear un punto de enlace de VPC en esas subredes para Amazon Redshift o Amazon Redshift sin servidor. De esta forma, el controlador JDBC puede obtener las credenciales.

Lectura y escritura desde y hacia Amazon Redshift

Los siguientes ejemplos de código utilizan PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con una API de origen de datos y con SparkSQL.

Data source API

Utilice PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con una API de origen de datos.

```
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

Utilice PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con SparkSQL.

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

bucket = "s3://path/for/temp/data"
tableName = "table-name" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {table-name} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{table-name}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws-iam-role-arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
```

```
df.write.insertInto(table-name, overwrite=False)
df = spark.sql(f"SELECT * FROM {table-name}")
df.show()
```

Consideraciones y limitaciones al utilizar el conector de Spark

- Le sugerimos que active SSL para la conexión JDBC desde Spark en Amazon EMR a Amazon Redshift.
- Le sugerimos que administre las credenciales del clúster de Amazon Redshift en AWS Secrets Manager como práctica recomendada. Consulte un ejemplo en Uso de AWS Secrets Manager para recuperar credenciales para una conexión a Amazon Redshift.
- Le sugerimos que pase un rol de IAM con el parámetro `aws_iam_role` para el parámetro de autenticación de Amazon Redshift.
- Actualmente, el parámetro `tempformat` no admite el formato Parquet.
- El URI `tempdir` apunta 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 sugerimos que bloquee el acceso público al clúster de Amazon Redshift.
  - Le sugerimos que active el registro de auditoría de Amazon Redshift.
  - Le sugerimos que active el cifrado en reposo de Amazon Redshift.
- Tenga en cuenta las siguientes recomendaciones para Amazon S3:
  - Le sugerimos que bloquee el acceso público a los buckets de Amazon S3.
  - Le sugerimos que utilice el cifrado del servidor de Amazon S3 para cifrar los buckets de Amazon S3 utilizados.
  - Le sugerimos 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 desde el código abierto a la imagen. Por motivos de seguridad, no admitimos los siguientes métodos de autenticación de Spark a Amazon S3:
    - Establecimiento de las claves de acceso de AWS en la clasificación de configuración `hadoop-env`
    - Codificación de las claves de acceso de AWS en el URI de `tempdir`

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 EMR sin servidor

En este tutorial, cargará un subconjunto de datos del United States Board on Geographic Names a un bucket de Amazon S3 y, a continuación, utilizará Hive o Spark en Amazon EMR sin servidor para copiar los datos en una tabla de Amazon DynamoDB que pueda consultar.

Paso 1: cargue los datos en un bucket de Amazon S3

Para crear un bucket de Amazon S3, siga las instrucciones en Crear un bucket en la Guía del usuario de la consola de Amazon Simple Storage Service. Sustituya las referencias a *`amzn-s3-demo-bucket`* por el nombre del bucket recién creado. Ahora su aplicación EMR sin servidor está lista para ejecutar trabajos.

1. Descargue el archivo de datos de ejemplo `features.zip` con el comando siguiente.

   ```
   wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
   features.zip
   ```

2. Extraiga el archivo `features.txt` del archivo y acceda a las primeras líneas del archivo:

   ```
   unzip features.zip
   head features.txt
   ```

   El resultado debería ser 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, nombre, tipo de característica natural, estado, latitud en grados, longitud en grados y altura en pies.

3. Descarga de datos en Amazon S3

```
aws s3 cp features.txt s3://amzn-s3-demo-bucket/features/
```

Paso 2: cree una tabla

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 comando siguiente.

```
import org.apache.spark.sql.SparkSession

val sparkSession = SparkSession.builder().enableHiveSupport().getOrCreate()

sparkSession.sql("CREATE TABLE hive_features \
    (feature_id BIGINT, \
    feature_name STRING, \
    feature_class STRING, \
    state_alpha STRING, \
    prim_lat_dec DOUBLE, \
    prim_long_dec DOUBLE, \
    elev_in_ft BIGINT) \
    ROW FORMAT DELIMITED \
    FIELDS TERMINATED BY '|' \
    LINES TERMINATED BY '\n' \
    LOCATION 's3://amzn-s3-demo-bucket/features';")
```

Ahora, tenemos una tabla de Hive con datos que contiene los datos del archivo `features.txt`. Para comprobar si sus datos están en la tabla, ejecute una consulta de Spark SQL 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, ejecute el comando siguiente.

```
CREATE TABLE hive_features
    (feature_id             BIGINT,
    feature_name            STRING ,
    feature_class           STRING ,
    state_alpha             STRING,
    prim_lat_dec            DOUBLE ,
    prim_long_dec           DOUBLE ,
    elev_in_ft              BIGINT)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '|'
    LINES TERMINATED BY '\n'
    LOCATION 's3://amzn-s3-demo-bucket/features';
```

Ahora tiene una tabla de Hive que contiene los datos del archivo `features.txt`. Para comprobar si sus 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: copie los datos a DynamoDB

Utilice 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://amzn-s3-demo-bucket/ddb-connector/")
            .map(row => {
                val line = row.split("\\|")
                val item = new DynamoDBItemWritable()

                val elevInFt = if (line.length > 6) {
                    new AttributeValue().withN(line(6))
                } else {
                    new AttributeValue().withNULL(true)
                }

                item.setItem(Map(
                    "feature_id" -> new AttributeValue().withN(line(0)),
                    "feature_name" -> new AttributeValue(line(1)),
                    "feature_class" -> new AttributeValue(line(2)),
                    "state_alpha" -> new AttributeValue(line(3)),
                    "prim_lat_dec" -> new AttributeValue().withN(line(4)),
                    "prim_long_dec" -> new AttributeValue().withN(line(5)),
                    "elev_in_ft" -> elevInFt)
                    .asJava)
                (new Text(""), item)
        })
        rdd.saveAsHadoopDataset(jobConf)
    }
```

```
}
```

Hive

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: haga una 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, use Spark SQL o la API MapReduce de Spark.

Example — Consulta de su tabla de DynamoDB con Spark SQL

La siguiente consulta de Spark SQL devuelve una lista de todos los tipos de características en orden alfabético.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

La siguiente consulta de Spark SQL 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 consulta de Spark SQL devuelve una lista de todos los estados con al menos tres características 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 de su tabla de DynamoDB con la API MapReduce de Spark

La siguiente consulta de MapReduce devuelve una lista de todos los tipos de características 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 consulta de MapReduce 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 consulta de MapReduce devuelve una lista de todos los estados con al menos tres características 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 DynamoDB que creó en el paso anterior, siga las instrucciones de Consultar los datos en la tabla de DynamoDB.

Configuración del acceso entre cuentas

Para configurar el acceso entre cuentas de EMR sin servidor, complete los siguientes pasos. En el ejemplo, `AccountA` es la cuenta en la que creó la aplicación de Amazon EMR sin servidor y `AccountB` es la cuenta en la que se encuentra su Amazon DynamoDB.

1. Cree una tabla DynamoDB en `AccountB`. Para obtener más información, consulte Paso 1: crear una tabla.
2. Cree un rol de IAM del `Cross-Account-Role-B` en la `AccountB` que pueda acceder a la tabla DynamoDB.

   a. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.

   b. Elija Roles y cree un nuevo rol llamado `Cross-Account-Role-B`. Para obtener más información acerca de cómo crear roles de IAM, consulte Creación de roles de IAM en la Guía del usuario.

   c. Cree una política de IAM que conceda permisos para acceder a la tabla DynamoDB entre cuentas. Adjunte la política de IAM al `Cross-Account-Role-B`.

      A continuación se muestra una política que concede acceso a una tabla DynamoDB `CrossAccountTable`.

   d. Edite la relación de confianza del rol `Cross-Account-Role-B`.

      Para configurar la relación de confianza del rol, elija la pestaña Relaciones de confianza en la consola de IAM para el rol que creó en Paso 2: Cross-Account-Role-B.

      Seleccione Editar relación de confianza y, a continuación, añada el siguiente documento de política. Este documento permite a `Job-Execution-Role-A` en `AccountA` asumir este rol `Cross-Account-Role-B`.

JSON

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

e. Conceda a `Job-Execution-Role-A` en `AccountA` permisos - `STS Assume role` para asumir `Cross-Account-Role-B`.

En la consola de IAM para la `AccountA` de Cuenta de AWS, 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`.

JSON

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

f. Establezca la propiedad `dynamodb.customAWSCredentialsProvider` con un valor como `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` en la clasificación de sitios principales. Establezca localmente la variable de entorno `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` con valor del ARN de `Cross-Account-Role-B`.

3. Ejecute el trabajo de Spark o Hive usando `Job-Execution-Role-A`.

Consideraciones

Tenga en cuenta estos comportamientos y limitaciones cuando utilice el conector de DynamoDB con Apache Spark o Apache Hive.

Consideraciones a la hora de utilizar el conector de DynamoDB con Apache Spark

- Spark SQL no admite la creación de una tabla Hive con la opción de controlador de almacenamiento. Para obtener más información, consulte Specifying storage format for Hive tables en la documentación de Apache Spark.
- Spark SQL no admite la operación `STORED BY` 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 inserción de predicados. La inserción de predicados filtra los datos por una columna que está asignada a la clave de partición de una tabla de DynamoDB. La inserción de predicados solo funciona cuando se utiliza el conector con Spark SQL y no con la API MapReduce.

Consideraciones a la hora de utilizar el conector de DynamoDB con Apache Hive

Ajuste del número máximo de mapeadores

- Si utiliza la consulta `SELECT` para leer datos de una tabla de Hive externa que se asigna en DynamoDB, el número de tareas de asignación en EMR sin servidor se calcula como el rendimiento de lectura total configurado para la tabla de DynamoDB, dividido por el rendimiento por tarea de asignación. El rendimiento predeterminado por tarea de asignación es 100.
- El trabajo de Hive puede utilizar un número de tareas de asignación 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 que tarde bastante en ejecutarse

puede consumir toda la capacidad de lectura aprovisionada de la tabla de DynamoDB. Esto afecta negativamente a otros usuarios.

- Puede usar la propiedad `dynamodb.max.map.tasks` para establecer un límite superior para las tareas de asignación. También puede utilizar esta propiedad para ajustar la cantidad de datos que lee cada tarea de asignación en función del tamaño del contenedor de tareas.
- Puede establecer la propiedad `dynamodb.max.map.tasks` en el nivel de consulta de Hive o en la clasificación `hive-site` del comando start-job-run. Este valor debe ser igual o superior a 1. Cuando Hive procesa la consulta, el trabajo de Hive resultante no usa más que los valores de `dynamodb.max.map.tasks` cuando realiza lecturas en la tabla de DynamoDB.

Ajuste del rendimiento de escritura por tarea

- El rendimiento de escritura por tarea en EMR sin servidor se calcula como el rendimiento de escritura total que se configura para una tabla de DynamoDB, dividido por el valor de la propiedad `mapreduce.job.maps`. Para 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 en otros trabajos.
- Para evitar la limitación de la escritura, establezca el valor de la propiedad `mapreduce.job.maps` 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 clasificación `mapred-site` del comando start-job-run en el EMR sin servidor.

Seguridad

La seguridad en la nube de AWS es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y el usuario. 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 ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Auditores independientes prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información sobre los programas de conformidad que se aplican a Amazon EMR sin servidor, consulte AWS services in scope by compliance program.
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. 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 permite comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon EMR sin servidor. Los temas explican cómo configurar Amazon EMR sin servidor y utilizar otros productos de AWS para satisfacer sus necesidades de seguridad y objetivos de conformidad.

Temas

- Validación de conformidad para Amazon EMR sin servidor
- Resiliencia de Amazon EMR sin servidor
- Seguridad de la infraestructura de Amazon EMR sin servidor
- Configuración y análisis de vulnerabilidades en Amazon EMR sin servidor

Prácticas recomendadas de seguridad para Amazon EMR sin servidor

Amazon EMR sin servidor 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

EMR sin servidor proporciona una política de acceso granular para las aplicaciones que utilizan roles de IAM, como los roles de ejecución. Sugerimos 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

EMR sin servidor crea un aislamiento total de la red entre los trabajos que pertenecen a diferentes aplicaciones EMR sin servidor. En los casos en que se desee el aislamiento a nivel de trabajo, considere la posibilidad de aislar los trabajos en diferentes aplicaciones EMR sin servidor.

Permisos de control de acceso basado en roles (RBAC)

Los administradores deben controlar estrictamente los permisos de control de acceso basado en roles (RBAC) para aplicaciones de EMR sin servidor.

Protección de los datos

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en Amazon EMR sin servidor. Tal como se describe en este modelo, AWS es responsable de proteger la

infraestructura global que ejecuta toda la nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Este contenido incluye la configuración de seguridad y las tareas de administración de los servicios de AWS que usted utiliza. Para obtener más información sobre la privacidad de datos, consulte Data Privacy FAQ. Para obtener información sobre la protección de datos en Europa, consulte la publicación del blog The AWS Shared Responsibility Model and GDPR en el blog de seguridad de AWS.

A los efectos de la protección de datos, sugerimos que proteja las credenciales de la cuenta de AWS y configure 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:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utiliza SSL/TLS para comunicarse con los recursos de AWS. Le sugerimos usar TLS 1.2 o una versión posterior.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de AWS.
- 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 sin servidor para cifrar datos en reposo y en tránsito.
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de línea de comandos o una API, utilice un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar federal de procesamiento de información 140-2 (FIPS 140-2).

Le sugerimos encarecidamente que nunca introduzca información de identificación confidencial, como números de cuenta de sus clientes, en los campos de formato libre, como el campo Nombre. Esto incluye los casos en los que trabaje con Amazon EMR sin servidor u otros servicios de AWS mediante la consola, la API, la AWS CLI o los SDK de AWS. Es posible que cualquier dato que introduzca en Amazon EMR sin servidor u otros servicios se incluya en los registros de diagnóstico. Cuando proporcione una URL a un servidor externo, no incluya información de credenciales en la URL para validar la solicitud para 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. Puede elegir entre varias opciones, incluidas claves administradas por AWS Key Management Service, claves administradas por Amazon S3, así como claves y certificados de proveedores personalizados que usted proporcione. 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 Precios de AWS KMS.

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 datos de EMRFS en Amazon S3

Cada aplicación EMR sin servidor utiliza una versión de lanzamiento específica, que incluye EMRFS (Sistema de archivos EMR). El cifrado de Amazon S3 funciona con objetos del sistema de archivos de EMR (EMRFS) que se leen y se escriben en Amazon S3. Puede especificar el cifrado del servidor (SSE) o el cifrado del cliente (CSE) de Amazon S3 como Modo de cifrado predeterminado al habilitar el cifrado en reposo. Opcionalmente, especifique métodos de cifrado diferentes para buckets individuales mediante Anulaciones de cifrado por bucket. Independientemente de si el cifrado de Amazon S3 está habilitado, la seguridad de la capa de transporte (TLS) cifra los objetos de EMRFS en tránsito entre los nodos del clúster de EMR y Amazon S3. Si utiliza Amazon S3 CSE con claves administradas por el cliente, su rol de ejecución utilizado para ejecutar trabajos en una aplicación EMR sin servidor debe tener acceso a la clave. Para obtener información más 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 utilice AWS KMS, se aplican cargos por el almacenamiento y el uso de las claves de cifrado. Para obtener más información, consulte Precios de AWS KMS.

Cifrado del servidor de Amazon S3

Todos los buckets de Amazon S3 tienen el cifrado configurado de forma predeterminada, y todos los objetos nuevos que se cargan en un bucket de S3 se cifran automáticamente en reposo. Amazon S3 cifra los datos del objeto a medida que los escribe en el disco y los descifra cuando se accede a ellos. Para obtener más información sobre SSE, consulte Protección de los datos con el cifrado del servidor en la Guía para desarrolladores de Amazon Simple Storage Service.

Puede elegir entre dos sistemas de administración de claves distintos al especificar SSE en Amazon EMR sin servidor:

- SSE-S3: Amazon S3 administra las claves en su nombre. No se requiere ninguna configuración adicional en EMR sin servidor.
- SSE-KMS: se utiliza una AWS KMS key para configurar políticas adecuadas destinadas a EMR sin servidor. No se requiere ninguna configuración adicional en EMR sin servidor.

Para usar el cifrado AWS KMS de los datos que escribe en Amazon S3, tiene dos opciones cuando usa la API de `StartJobRun`. Puede habilitar el cifrado de todo lo que escriba en Amazon S3 o habilitar el cifrado de los datos que escriba en un bucket específico. Para obtener más información sobre la API de `StartJobRun`, consulte la EMR Serverless API Reference.

Para activar el cifrado AWS KMS de todos los datos que escriba en Amazon S3, utilice los siguientes comandos cuando llame a la API de `StartJobRun`.

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Para activar el cifrado AWS KMS de todos los datos que escriba en un bucket específico, utilice los siguientes comandos cuando llame a la API de `StartJobRun`.

```
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-bucket1>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-
bucket1>.serverSideEncryption.kms.keyId=<kms-id>
```

El SSE con claves proporcionadas por el cliente (SSE-C) no está disponible para su uso con EMR sin servidor.

Cifrado del cliente de Amazon S3

Con el cifrado del cliente de Amazon S3, el proceso de cifrado y descifrado de Amazon S3 se produce en el cliente de EMRFS disponible en cualquier aplicación 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 usar claves proporcionadas por AWS KMS (CSE-KMS) o una clase de Java personalizada que proporciona la clave raíz del cliente (CSE-C). Los detalles de cifrado son ligeramente diferentes entre CSE-KMS y CSE-C, en función del proveedor especificado y de los metadatos del objeto que se descifra o se cifra. Si utiliza Amazon S3 CSE con claves administradas por el cliente, su rol de ejecución utilizado para ejecutar trabajos en una aplicación EMR sin servidor debe tener acceso a la clave. Podrían aplicarse cargos adicionales de KMS. Para obtener más información sobre estas diferencias, consulte Protección de los datos con el cifrado 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 de cifrado AES-256 estándar del sector.

Administración de claves

Puede configurar KMS para que rote automáticamente las claves de 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 más información, consulte Rotación de claves administradas por el cliente.

Cifrado en tránsito

Las siguientes características de cifrado específicas de la aplicación están disponibles con Amazon EMR sin servidor:

- Spark
  - De forma predeterminada, la comunicación entre los controladores y los ejecutores de Spark está autenticada y es interna. La comunicación RPC entre los controladores y los ejecutores está cifrada.
- Hive
  - La comunicación entre el metaalmacén de Glue de AWS y las aplicaciones EMR sin servidor se realiza a través de TLS.

Debería permitir solo las conexiones cifradas a través de HTTPS (TLS) mediante la condición aws:SecureTransport en las políticas de IAM del bucket de Amazon S3.

Identity and Access Management (IAM) en Amazon EMR sin servidor

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda a los gestionadores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon EMR sin servidor. IAM es un servicio de Servicio de AWS que se puedes utilizar sin cargo adicional.

Temas

Público

La forma en que utiliza AWS Identity and Access Management (IAM) varía en función del rol:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de identidad y acceso de Amazon EMR sin servidor)
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Identity and Access Management (IAM) en Amazon EMR sin servidor)

- Administrador de IAM: escriba políticas para administrar el acceso (consulte Políticas basadas en identidades de ejemplo para EMR sin servidor)

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado como Usuario raíz de la cuenta de AWS, un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de un origen de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales de Google/Facebook. Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In.

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Usuario raíz de la Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión llamada Cuenta de AWS del usuario raíz que tiene acceso completo a todos los Servicios de AWS y recursos. Recomendamos encarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidades para acceder a Servicios de AWS con credenciales temporales.

Una identidad federada es un usuario del directorio empresarial, proveedor de identidades web o Directory Service que accede a Servicios de AWS mediante credenciales de un origen de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos AWS IAM Identity Center. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales a largo plazo. Para obtener más información, consulte Solicitar que los usuarios humanos utilicen la federación con un proveedor de identidades para acceder a AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un usuario a un rol de IAM (consola) o llamando a una operación de la API de la AWS CLI o AWS. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuarios federados, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política define permisos cuando se asocia con una identidad o recurso. AWS evalúa estas políticas cuando una entidad principal realiza una solicitud. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Mediante políticas, los administradores especifican quién tiene acceso a qué definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a los roles, que luego los usuarios pueden asumir. Las políticas de IAM definen permisos independientemente del método utilizado para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos JSON de políticas de permisos que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en la identidad pueden ser políticas integradas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las políticas de confianza de roles de IAM 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. Debe especificar una entidad principal en una política en función de recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas por AWS en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer el máximo de permisos concedidos por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en la identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations.
- Políticas de control de recursos (RCP): establecen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations.

- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

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 acerca de cómo AWS decide si permitir o no una solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona EMR sin servidor con IAM

Antes de utilizar IAM para administrar el acceso a Amazon EMR sin servidor, obtenga información sobre qué características de IAM se encuentran disponibles con Amazon EMR sin servidor.

Características de IAM que puede utilizar con EMR sin servidor

Característica de IAM	Compatibilidad de Amazon EMR sin servidor
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	No
ACL	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	No
Roles vinculados al servicio	Sí

Para obtener una perspectiva general sobre cómo funcionan EMR sin servidor y otros productos de AWS con la mayoría de las características de IAM, consulte Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Políticas basadas en identidades de EMR sin servidor

Compatibilidad con las políticas basadas en identidad: sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, 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 más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Políticas basadas en identidades de ejemplo para EMR sin servidor

Para acceder a ejemplos de políticas basadas en identidades de Amazon EMR sin servidor, consulte Ejemplos de políticas basadas en identidades para EMR sin servidor.

Políticas basadas en recursos dentro de Amazon EMR sin servidor

Admite políticas basadas en recursos: no

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM 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. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Para habilitar el acceso entre cuentas, puedes especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Acciones de políticas para EMR sin servidor

Compatibilidad con las acciones de políticas: sí

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

A fin de conocer una lista completa de acciones de EMR sin servidor, consulte Actions, resources, and condition keys for Amazon EMR Serverless en la Referencia de autorizaciones de servicio.

Las acciones de políticas de EMR sin servidor 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 acceder a ejemplos de políticas basadas en identidades de Amazon EMR sin servidor, consulte Ejemplos de políticas basadas en identidades para EMR sin servidor.

Recursos de políticas para EMR sin servidor

Compatibilidad con los recursos de políticas: sí

Los administradores pueden utilizar las políticas JSON de AWS 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` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Para las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de tipos de recursos de Amazon EMR sin servidor y sus ARN, consulte Resources defined by Amazon EMR Serverless en la Referencia de autorizaciones de servicio. Para saber con qué acciones especifican el ARN de cada recurso, consulte Actions, resources, and condition keys for Amazon EMR Serverless.

Para acceder a ejemplos de políticas basadas en identidades de Amazon EMR sin servidor, consulte Ejemplos de políticas basadas en identidades para EMR sin servidor.

Claves de condición de políticas para EMR Serverless

Compatibilidad con claves de condición de políticas

Admite claves de condición de políticas específicas del servicio	No

Los administradores pueden utilizar las políticas JSON de AWS 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` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puedes crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Para obtener una lista de las claves de condición de Amazon EMR sin servidor y para más información sobre qué acciones y recursos admiten el uso de una clave de condición, consulte Actions, resources, and condition keys for Amazon EMR Serverless en la Referencia de autorizaciones de servicio.

Todas las acciones de Amazon EC2 admiten las claves de condición `aws:RequestedRegion` y `ec2:Region`. Para obtener más información, consulte Ejemplo: Restringir el acceso a una región específica.

Listas de control de acceso (ACL) en EMR sin servidor

Compatibilidad con ACL: no

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Control de acceso basado en atributos (ABAC) con EMR sin servidor

Compatibilidad con el control de acceso basado en atributos (ABAC)

Admite ABAC (etiquetas en las políticas)	Sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos denominados etiquetas. Puede asociar etiquetas a entidades de IAM y recursos de AWS y, a continuación, diseñar políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso.

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 sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Uso de credenciales temporales con EMR sin servidor

Compatibilidad con credenciales temporales: sí

Las credenciales temporales proporcionan acceso a corto plazo a los recursos de AWS y se crean automáticamente cuando se utiliza la federación o se cambia de rol. AWS recomienda que genere

de forma dinámica credenciales temporales en lugar de usar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM y Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Permisos de entidades principales entre servicios de EMR sin servidor

Admite sesiones de acceso directo (FAS): sí

Las sesiones de acceso directo (FAS) utilizan los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Roles de servicio para EMR sin servidor

Compatible con roles de servicio	No

Roles vinculados al servicio de EMR sin servidor

Compatible con roles vinculados al servicio	Sí

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para acceder a la documentación acerca del rol vinculado a servicios para ese servicio.

Uso de rRoles vinculados al servicio de EMR sin servidor

Amazon EMR sin servidor utiliza roles vinculados a un servicio de AWS Identity and Access Management (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a un EMR sin servidor. Los roles vinculados a servicios son predefinidos por EMR sin servidor e incluyen todos los permisos que el servicio requiere para llamar a otros servicios de AWS en su nombre.

Un rol vinculado a un servicio simplifica la configuración de EMR sin servidor porque ya no tendrá que añadir manualmente los permisos necesarios. EMR sin servidor define los permisos de sus roles

vinculados a los servicios y, a menos que esté definido de otra manera, solo EMR sin servidor puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda adjuntar a ninguna otra entidad de IAM.

Solo es posible eliminar un rol vinculado a un servicio después de eliminar sus recursos relacionados. De esta forma se protegen los recursos de EMR sin servidor, ya que evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información sobre otros servicios que admiten roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Sí en la columna Roles vinculados a servicios. Elija una opción Sí con un enlace para acceder a la documentación acerca del rol vinculado al servicio en cuestión.

Permisos de rol vinculados a servicios para EMR sin servidor

EMR sin servidor utiliza el rol vinculado a servicio denominado AWSServiceRoleForAmazonEMRServerless para permitir que llame a las API de AWS en su nombre.

El rol vinculado al servicio AWSServiceRoleForAmazonEMRServerless confía en que los siguientes servicios asuman el rol:

- `ops.emr-serverless.amazonaws.com`

La política de permisos del rol denominada `AmazonEMRServerlessServiceRolePolicy` permite que EMR sin servidor realice las siguientes acciones en los recursos especificados:

> **Note**
>
> El contenido de las políticas administradas cambia, por lo que es posible que la política que mostramos aquí se haya quedado obsoleta. Puede ver la versión más actualizada de AmazonEMRServerlessServiceRolePolicy en la Consola de administración de AWS.

- 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 política es la política `AmazonEMRServerlessServiceRolePolicy` completa.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EC2PolicyStatement",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeRouteTables"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CloudWatchPolicyStatement",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
```

```
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/EMRServerless",
            "AWS/Usage"
          ]
        }
      }
    }
  ]
}
```

Para permitir que el servicio principal de EMR sin servidor asuma este rol, adjunte la siguiente política de confianza al rol.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/aws-service-role/emr-
serverless.amazonaws.com/AWSServiceRoleForEMRServerless",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio para EMR sin servidor

No necesita crear manualmente un rol vinculado a servicios. Cuando cree una nueva aplicación de EMR sin servidor en la Consola de administración de AWS (mediante EMR Studio), el AWS CLI o la

API de AWS, EMR sin servidor crea el rol vinculado al servicio por usted. Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios.

Para crear un rol vinculado a un servicio AWSServiceRoleForAmazonEMRServerless con IAM

Agregue la siguiente instrucción a la política de permisos de la entidad de IAM que tiene que 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 de EMR sin servidor, este se encarga nuevamente de crear el rol vinculado a servicios.

Puede utilizar la consola de IAM para crear un rol vinculado a servicios con el caso de uso EMR sin servidor. En la AWS CLI o la API de AWS, cree un rol vinculado al servicio con el nombre de servicio `ops.emr-serverless.amazonaws.com`. Para obtener más información, consulte Creación de un rol vinculado a un servicio en la Guía del usuario de 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 EMR sin servidor

EMR sin servidor no permite editar el rol vinculado al servicio AWSServiceRoleForAmazonEMRServerless. No puede editar la política de IAM de propiedad de AWS que utiliza el rol vinculado al servicio de EMR sin servidor ya que contiene todos los permisos que necesita EMR sin servidor. Sin embargo, puede editar la descripción del rol mediante IAM.

Para editar la descripción del rol vinculado a un servicio AWSServiceRoleForAmazonEMRServerless con IAM

Agregue la siguiente instrucción a la política de permisos de la entidad de IAM que tiene que editar la descripción del 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 Cómo editar un rol vinculado a un servicio en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio para EMR sin servidor

Si ya no necesita usar una característica o un servicio que requieran un rol vinculado a un servicio, le sugerimos que elimine dicho rol. De esta forma no tiene una entidad no utilizada que no se supervise ni mantenga de forma activa. Sin embargo, debe eliminar todas las aplicaciones de EMR sin servidor de todas las regiones para poder eliminar el rol vinculado a servicio.

> **Note**
>
> Si el servicio EMR sin servidor está utilizando el rol cuando intenta eliminar los recursos asociados al rol, la eliminación podría producir un error. En tal caso, espere unos minutos e intente de nuevo la operación.

Para eliminar un rol vinculado a un servicio AWSServiceRoleForAmazonEMRServerless con IAM

Agregue la siguiente instrucción a la política de permisos de la entidad de IAM que tiene que 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 a servicios mediante IAM

Utilice la consola de IAM, la AWS CLI o la API de AWS para eliminar el rol vinculado a servicios AWSServiceRoleForAmazonEMRServerless. Para obtener más información, consulte Cómo eliminar un rol vinculado a servicios en la Guía del usuario de IAM.

Regiones admitidas para los roles vinculados a un servicio de EMR sin servidor

EMR sin servidor admite el uso de roles vinculados a servicios en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte Regiones y puntos de conexión de AWS.

Roles en tiempo de ejecución de trabajo para Amazon EMR sin servidor

Puede especificar los permisos del rol de IAM que una ejecución de trabajo de EMR sin servidor puede asumir cuando llame a otros servicios en su nombre. Esto incluye el acceso a Amazon S3 para todos las fuentes de datos, los destinos y otros recursos de AWS, como los clústeres de Amazon Redshift y las tablas de DynamoDB. Para obtener más información acerca de cómo crear un rol, consulte Crear un rol de tiempo de ejecución del trabajo.

Políticas de tiempo de ejecución de ejemplo

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 trabajos permite:

- Acceso de lectura a los buckets de Amazon S3 con ejemplos de EMR.
- Acceso completo a los buckets de S3.
- Acceso de creación y lectura al catálogo de datos de Glue de AWS.

Para añadir acceso a otros recursos de AWS, como DynamoDB, tendrá que incluir sus permisos en la política al crear el rol de tiempo de ejecución.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.elasticmapreduce",
        "arn:aws:s3:::*.elasticmapreduce/*"
      ]
    },
    {
      "Sid": "FullAccessToS3Bucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
```

```
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Transferencia de los privilegios del rol

Puede asociar políticas de permisos de IAM al rol de usuario para permitir al usuario que transfiera solo los roles aprobados. Esto permite a los administradores controlar qué usuarios pueden transferir roles específicos de tiempo de ejecución de trabajos a trabajos de EMR sin servidor. Para obtener más información sobre los permisos de configuración, consulte Concesión de permisos a un usuario para transferir un rol a un servicio de AWS.

El siguiente es un ejemplo de política que permite transferir un rol de tiempo de ejecución de un trabajo a la entidad principal del servicio EMR sin servidor.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

Políticas de permisos administradas asociadas con las funciones del tiempo de ejecución

Cuando envía ejecuciones de trabajos a EMR sin servidor a través de la consola de EMR Studio, hay un paso en el que debe elegir un rol de tiempo de ejecución para asociarlo a su aplicación. Hay

políticas administradas subyacentes asociadas a cada selección de la consola que debe tener en cuenta. Las tres opciones son las siguientes:

1. Todos los buckets: esta opción especifica la política administrada de AWS Amazons3FullAccess, lo que proporciona acceso completo a todos los buckets.
2. Buckets específicos: esta opción especifica el identificador del nombre de recurso de Amazon (ARN) de cada bucket que seleccione. No hay una política administrada subyacente.
3. Ninguno: no se incluye ningún permiso de política administrada.

Le sugerimos que agregue buckets específicos. Si elige todos los buckets, tenga en cuenta que esta opción configura el acceso completo a todos los buckets.

Ejemplos de políticas de acceso de usuario para EMR sin servidor

Puede configurar políticas detalladas para sus usuarios en función de las acciones que desee que realice cada usuario al interactuar con las aplicaciones EMR sin servidor. Las siguientes políticas son ejemplos que pueden ayudar a configurar los permisos adecuados 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 Configure EMR Studio user permissions. Para obtener información sobre cómo asociar políticas a los usuarios de IAM (entidades principales), consulte Administración de políticas de IAM en la Guía del usuario de IAM.

Política de usuarios avanzados

Para conceder todas las acciones necesarias para EMR sin servidor, cree y adjunte una política de `AmazonEMRServerlessFullAccess` al usuario, rol o grupo de IAM 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 trabajos. Revela todas las acciones que EMR sin servidor requiere para otros servicios.

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication",
        "emr-serverless:UpdateApplication",
        "emr-serverless:DeleteApplication",
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StopApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Cuando habilita la conectividad de red con su VPC, las aplicaciones EMR sin servidor crean interfaces de red elásticas (ENI) de Amazon EC2 para comunicarse con los recursos de la VPC. La siguiente política garantiza que cualquier ENI de EC2 nuevo solo se cree en el contexto de las aplicaciones EMR sin servidor.

> **Note**
>
> Recomendamos encarecidamente establecer esta política para garantizar que los usuarios no puedan crear ENI de EC2 excepto en el contexto del lanzamiento de aplicaciones EMR sin servidor.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
```

```
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
        }
      }
    }
  ]
}
```

Si desea restringir el acceso de EMR sin servidor a determinadas subredes, puede etiquetar cada subred con una condición de etiqueta. Esta política de IAM garantiza que las aplicaciones de EMR sin servidor solo puedan crear ENI de EC2 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 crea su primera aplicación, debe configurar sus políticas de permisos para que le permitan crear un rol vinculado al servicio

EMR sin servidor. Para obtener más información, consulte Uso de rRoles vinculados al servicio de EMR sin servidor.

La siguiente política de IAM le permite crear un rol vinculado al servicio de 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 ingeniero de datos

A continuación, se muestra un ejemplo de política que permite a los usuarios permisos de solo lectura en las 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.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
```

```
      ]
    }
  ]
}
```

Uso de etiquetas para el control de acceso

Puede usar condiciones de etiquetado para el control de acceso detallado. Por ejemplo, puede restringir a los usuarios de un equipo para que solo puedan enviar trabajos a aplicaciones EMR sin servidor y etiquetadas con el nombre de su equipo.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

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 a las ejecuciones de trabajo basadas en etiquetas.

Los siguientes ejemplos muestran distintos supuestos y formas de utilizar los operadores de condiciones con las claves de condición de EMR sin servidor. Estas instrucciones de política de IAM tienen fines demostrativos 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 la planificación y las pruebas de las políticas de IAM, consulte la Guía del usuario de IAM.

> **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 que se muestran a continuación muestran las políticas de permisos basadas en identidades que se utilizan para controlar las acciones que se permiten con las aplicaciones de EMR sin servidor.

Permitir acciones solo en recursos con valores de etiqueta específicos

En el siguiente ejemplo de política, la condición `StringEquals` intenta hacer coincidir `dev` con el valor de la etiqueta Department. Si la etiqueta Department no se agregó a la aplicación o no contiene el valor `dev`, la política no se aplica y esta política no permite las acciones. Si no hay otras instrucciones de política que permitan las acciones, el usuario solo puede trabajar con aplicaciones que tengan este valor para esta etiqueta.

JSON

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

```
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSGetapplication"
    }
  ]
}
```

También puede especificar varios valores de etiqueta utilizando un operador de condición. Por ejemplo, para permitir acciones en las aplicaciones donde la etiqueta `department` contenga el valor `dev` o `test`, reemplace el bloque de condición en el ejemplo anterior por los siguientes.

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

Requerir etiquetado cuando se crea un recurso

En el siguiente ejemplo, la etiqueta debe aplicarse al crear la aplicación.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
```

```
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

La siguiente instrucción de política permite a un usuario crear una aplicación solo si la aplicación tiene una etiqueta `department`, que puede contener cualquier valor.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-east-1", "us-west-2"]
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

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 etiqueta `department` cuyo valor no sea `dev`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSTagresource"
    }
  ]
}
```

Ejemplos de políticas basadas en identidades para EMR sin servidor

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Amazon EMR sin servidor. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Amazon EMR sin servidor, incluido el formato de los ARN para cada uno de los tipos de recursos, consulte Acciones, recursos y claves de condición para Amazon EMR sin servidor en la Referencia de autorizaciones de servicio.

Temas

- Prácticas recomendadas sobre las políticas
- Cómo permitir a los usuarios acceder a sus propios permisos

Prácticas recomendadas sobre las políticas

> **Note**
>
> EMR sin servidor no admite políticas administradas, por lo que no se aplica la primera práctica que se detalla a continuación.

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon EMR sin servidor de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comienza con las políticas administradas por AWSy continúa con los permisos de privilegio mínimo: a fin de comenzar a conceder permisos a los usuarios y las cargas de tarea, utiliza las políticas administradas por AWS, que conceden permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Se recomienda definir políticas gestionadas por el cliente de AWS específicas para sus casos de uso a fin de reducir aún más los permisos. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, 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. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puedes escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado como, por ejemplo, CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesite usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para obtener una mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Cómo permitir a los usuarios acceder a sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Actualizaciones de Amazon EMR sin servidor a políticas administradas por AWS

Es posible acceder a los detalles sobre las actualizaciones de las políticas administradas de AWS para Amazon EMR sin servidor desde que este servicio comenzó a hacer un seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página del Historial de documentos de Amazon EMR sin servidor.

Cambio	Descripción	Fecha
AmazonEMRServerlessServiceRolePolicy: actualización de una política existente	Amazon EMR sin servidor agregó la nueva `Sid` `CloudWatchPolicyStatement` y `EC2PolicyStatement` a la Política AmazonEMRServerlessServiceRolePolicy.	25 de enero de 2024
AmazonEMRServerlessServiceRolePolicy: actualización de una política existente	Amazon EMR sin servidor agregó nuevos permisos para permitir que Amazon EMR	20 de abril de 2023

Cambio	Descripción	Fecha
	sin servidor publique métricas de cuenta agregadas para el uso de vCPU en el espacio de nombres `"AWS/Usage"` .	
Amazon EMR sin servidor ha comenzado a hacer un seguimiento de los cambios	Amazon EMR sin servidor comenzó a hacer un seguimiento de los cambios en sus políticas administradas por AWS.	20 de abril de 2023

Solución de problemas de identidad y acceso de Amazon EMR sin servidor

Utilice la siguiente información para diagnosticar y solucionar los problemas habituales que pueden surgir cuando se trabaja con Amazon EMR sin servidor.

Temas

- No tengo autorización para realizar una acción en Amazon EMR sin servidor
- No tengo autorización para realizar la operación iam:PassRole
- Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de Amazon EMR sin servidor.
- No es posible abrir el servidor del historial de Live UI/Spark desde EMR Studio para depurar el trabajo o se produce un error de API cuando se intenta obtener registros mediante get-dashboard-for-job-run

No tengo autorización para realizar una acción en Amazon EMR sin servidor

Si la Consola de administración de AWS le indica que no está autorizado para llevar a cabo una acción, entonces póngase en contacto con su administrador para obtener ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

El siguiente ejemplo de error se produce cuando el usuario `mateojackson` intenta utilizar la consola para acceder a 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 tengo autorización para realizar la operación iam:PassRole

Si recibe un error que indica que no tiene autorización para llevar a cabo la acción `iam:PassRole`, las políticas se deben actualizar para permitirle pasar un rol a Amazon EMR sin servidor.

Algunos Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Amazon EMR sin servidor. 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 gestionador de AWS. El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de Amazon EMR sin servidor.

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 admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon EMR sin servidor admite estas características, consulte Identity and Access Management (IAM) en Amazon EMR sin servidor.
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte Cómo proporcionar acceso a un usuario de IAM a otra Cuenta de AWS de la que es propietario en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a tus recursos a Cuentas de AWS de terceros, consulta Proporcionar acceso a Cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

No es posible abrir el servidor del historial de Live UI/Spark desde EMR Studio para depurar el trabajo o se produce un error de API cuando se intenta obtener registros mediante **`get-dashboard-for-job-run`**

Si utiliza el almacenamiento administrado de EMR sin servidor para los registros y su aplicación de EMR sin servidor se encuentra en una subred privada con puntos de conexión de VPC para Amazon S3 y adjunta una política de puntos de conexión para controlar el acceso, agregue los permisos que se mencionan en Registros para EMR sin servidor con almacenamiento administrado de su política de VPC al punto de conexión de la puerta de enlace de S3 para que EMR sin servidor almacene y proporcione los registros de la aplicación.

Propagación de identidades de confianza con Amazon EMR sin servidor

Con Amazon EMR 7.8.0 y versiones posteriores, puede propagar las identidades de los usuarios de AWS IAM Identity Center a cargas de trabajo interactivas con EMR sin servidor a través del punto de conexión de Apache Livy. Las cargas de trabajo interactivas de Apache Livy propagan aún más la identidad proporcionada a los servicios posteriores, Amazon S3, Lake Formation y Amazon Redshift, lo que permite un acceso seguro a los datos mediante la identidad del usuario en los servicios posteriores. En las siguientes secciones, se proporciona la descripción general conceptual,

los requisitos previos y los pasos necesarios para lanzar y propagar la identidad a cargas de trabajo interactivas con EMR sin servidor a través de un punto de conexión de Apache Livy.

Descripción general

IAM Identity Center es el enfoque recomendado para la autenticación y autorización de los empleados en AWS para organizaciones de cualquier tamaño y tipo. Con Identity Center, puede crear y administrar identidades de usuario en AWS o conectar una fuente de identidad existente, entre las que se incluyen Microsoft Active Directory, Okta, Ping Identity, JumpCloud, Google Workspace y Microsoft Entra ID (anteriormente Azure AD).

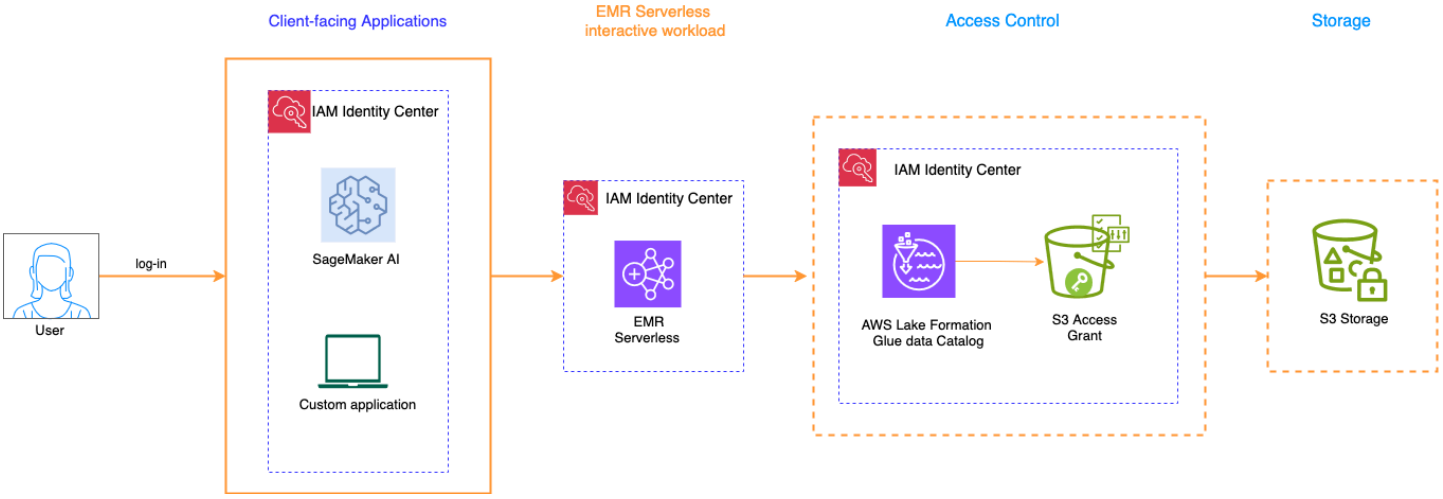
La propagación de identidades de confianza es una característica de AWS IAM Identity Center que los administradores de los servicios conectados de AWS pueden utilizar para conceder y auditar el acceso a los datos del servicio. El acceso a estos datos se basa en los atributos del usuario, como las asociaciones de grupo. La configuración de la propagación de identidades de confianza requiere la colaboración entre los administradores de los servicios conectados de AWS y los administradores de IAM Identity Center. Para obtener más información, consulte Prerequisites and considerations de la Guía del usuario del IAM Identity Center.

Características y ventajas

La integración del punto de conexión de Apache Livy de EMR sin servidor con la propagación de identidades de confianza del IAM Identity Center ofrece las siguientes ventajas:

- La capacidad de aplicar la autorización a nivel de tabla con las identidades del Identity Center en las tablas de Catálogo de datos de AWS Glue administradas por AWS Lake Formation.
- La capacidad de imponer la autorización con las identidades de Identity Center en los clústeres de Amazon Redshift.
- Permite el seguimiento integral de las acciones de los usuarios para la auditoría.
- La capacidad de aplicar la autorización a nivel de prefijo de Amazon S3 con las identidades de Identity Center en los prefijos de S3 administrados por S3 Access Grants.

Funcionamiento



Ejemplo de casos de uso

Preparación de datos e ingeniería de características

Los científicos de datos de varios equipos de investigación colaboran en proyectos complejos mediante una plataforma de datos unificada. Inician sesión en Sagemaker AI con sus credenciales corporativas, lo que les permite acceder de manera inmediata a un enorme lago de datos compartido que abarca varias cuentas de AWS. A medida que comienza la ingeniería de características para los nuevos modelos de machine learning, las sesiones de Spark ejecutadas a través de EMR sin servidor refuerzan las políticas de seguridad a nivel de columnas y filas de Lake Formation en función de sus identidades propagadas. Los científicos pueden preparar los datos y diseñar características de manera eficiente con herramientas conocidas, mientras que los equipos de cumplimiento tienen la seguridad de que cada interacción de datos se rastrea y audita automáticamente. Este entorno seguro y colaborativo acelera los procesos de investigación y, al mismo tiempo, mantiene los estándares de protección de datos estrictos que se exigen en los sectores regulados.

Introducción a la propagación de identidades de confianza para EMR sin servidor

En esta sección, se brinda información sobre cómo configurar la aplicación de EMR sin servidor con el punto de conexión de Apache Ivy para permitir la integración con AWS IAM Identity Center y habilitar la propagación de identidades de confianza.

Requisitos previos

- Una instancia de Identity Center en la región de AWS en la que desea crear un punto de conexión de Apache Livy de EMR sin servidor para la propagación de identidades de confianza. Una instancia de Identity Center solo puede existir en una sola región para una cuenta de AWS. Consulte Enable IAM Identity Center y Cómo agregar usuarios y grupos de su fuente de identidades al IAM Identity Center.
- Habilite la propagación de identidades de confianza para los servicios posteriores como Lake Formation, S3 Access Grants o el clúster de Amazon Redshift con el que la carga de trabajo interactiva interactúa para acceder a los datos.

Permisos necesarios para crear una aplicación de EMR sin servidor para la propagación de identidades de confianza.

Además de los permisos básicos necesarios para acceder a EMR sin servidor, debe configurar permisos adicionales para la identidad o rol de IAM que utilice para crear una aplicación de EMR sin servidor para la propagación de identidades de confianza. Para la propagación de identidades de confianza, EMR sin servidor crea o impulsa una aplicación de Identity Center administrada por un servicio en su cuenta que se utiliza para validar y propagar las identidades en los servicios posteriores.

```
"sso:DescribeInstance",
"sso:CreateApplication",
"sso:DeleteApplication",
"sso:PutApplicationAuthenticationMethod",
"sso:PutApplicationAssignmentConfiguration",
"sso:PutApplicationGrant",
"sso:PutApplicationAccessScope"
```

- `sso:DescribeInstance`: otorga permiso para describir y validar instanceArn de IAM Identity Center que especifique en el parámetro de configuración de Identity Center.
- `sso:CreateApplication`: otorga permiso para crear una aplicación del IAM Identity Center administrada por EMR sin servidor que se utilice para acciones de propagación de identidades de confianza.
- `sso:DeleteApplication`: otorga permiso para limpiar una aplicación de IAM Identity Center administrada por EMR sin servidor.

- `sso:PutApplicationAuthenticationMethod`: otorga permiso para colocar authenticationMethod en la aplicación de IAM Identity Center administrada por EMR sin servidor, lo que le permite a la entidad principal de servicio de EMR sin servidor interactuar con la aplicación de IAM Identity Center.
- `sso:PutApplicationAssignmentConfiguration`: otorga permiso para establecer la configuración “User-assignment-not-required” en la aplicación de IAM Identity Center.
- `sso:PutApplicationGrant`: otorga permiso para aplicar el intercambio de tokens, introspectToken, refreshToken y revokeToken en la aplicación de IAM Identity Center.
- `sso:PutApplicationAccessScope`: otorga permiso para aplicar un alcance posterior que permita la propagación de identidades de confianza a la aplicación IAM Identity Center. Aplicamos “redshift:connect”, “lakeformation:query” y “s3:read_write” para habilitar la propagación de identidades de confianza a estos servicios.

Creación de una aplicación de EMR sin servidor para la propagación de identidades de confianza.

Se debe especificar el campo `–identity-center-configuration` con `identityCenterInstanceArn` para habilitar la propagación de identidades de confianza en la aplicación. Utilice el siguiente comando de ejemplo para crear una aplicación de EMR sin servidor que tenga habilitada la propagación de identidades de confianza.

> **Note**
>
> También debe especificar `--interactive-configuration '{"livyEndpointEnabled":true}'`, ya que la propagación de identidades de confianza está habilitada únicamente para el punto de conexión de Apache Livy.

```
aws emr-serverless create-application \
  --release-label emr-7.8.0 \
  --type "SPARK" \
  --identity-center-configuration '{"identityCenterInstanceArn" :
 "arn:aws:sso:::instance/ssoins-123456789"}' \
  --interactive-configuration '{"livyEndpointEnabled":true}'
```

- `identity-center-configuration` (opcional): habilita la propagación de identidades de confianza de Identity Center si se especifica.

- `identityCenterInstanceArn`: (obligatorio) el ARN de la instancia de Identity Center.

En caso de que no tenga los permisos de Identity Center necesarios (mencionados anteriormente), primero cree la aplicación de EMR sin servidor sin la propagación de identidades de confianza (por ejemplo, no especifique el parámetro `–identity-center-configuration`) y, luego, solicítele al administrador de Identity Center que habilite la propagación de identidades de confianza mediante la invocación de la API update-application. Vea un ejemplo a continuación:

```
aws emr-serverless update-application \
  --application-id applicationId \
  --identity-center-configuration '{"identityCenterInstanceArn" :
 "arn:aws:sso:::instance/ssoins-123456789"}'
```

EMR sin servidor crea una aplicación de Identity Center administrada por un servicio en su cuenta que se utiliza para la validación y la propagación de identidades en los servicios posteriores. La aplicación de Identity Center creada y administrada por EMR sin servidor se comparte en todas las aplicaciones de EMR sin servidor de confianza habilitadas para la propagación de identidades de confianza de su cuenta.

> **Note**
>
> No modifique manualmente la configuración de la aplicación de Identity Center administrada. Cualquier cambio podría afectar a todas las aplicaciones de EMR sin servidor habilitadas para la propagación de identidades de confianza en su cuenta.

Permisos del rol de ejecución del trabajo para propagar identidades

Dado que EMR sin servidor utiliza las credenciales del rol de ejecución de trabajos con identidad mejorada para propagar las identidades a los servicios posteriores AWS, la política de confianza del rol de ejecución del trabajo debe contar con un permiso adicional `sts:SetContext` para mejorar la credencial del rol de ejecución del trabajo con identidades a fin de permitir la propagación de identidades de confianza a los servicios posteriores, como la concesión de acceso a S3, Lake Formation o Amazon Redshift. Para obtener más información acerca de cómo crear un rol, consulte Create a job runtime role.

JSON

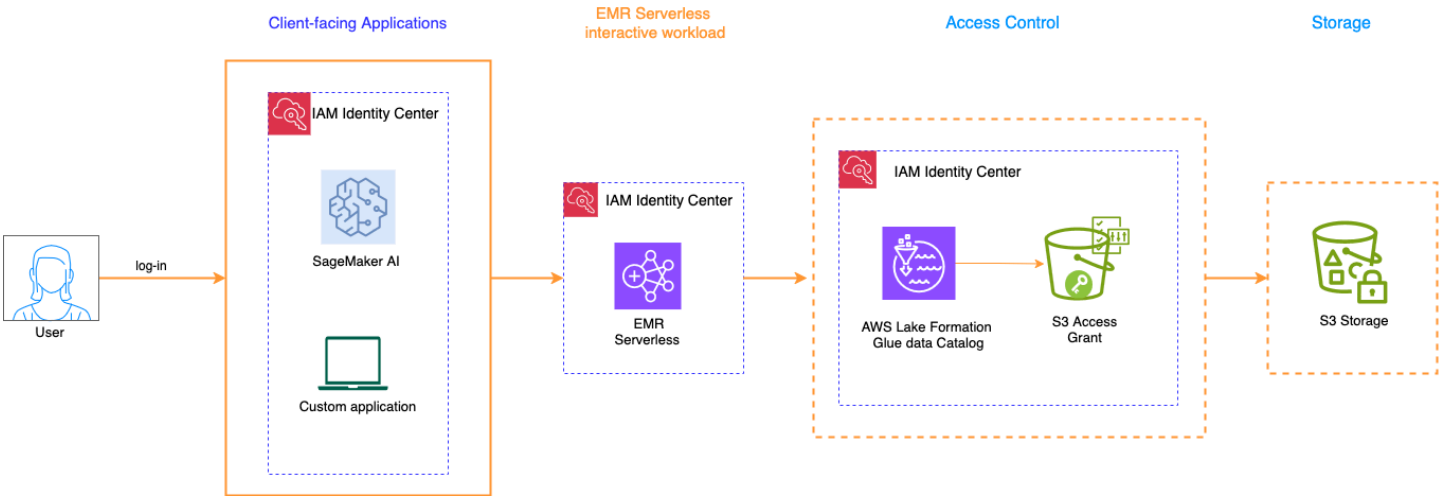
```
{
  "Version":"2012-10-17",
  "Statement": [
     {
     "Effect": "Allow",
     "Principal": {
     "Service": "emr-serverless.amazonaws.com"
     },
     "Action": [ "sts:AssumeRole", "sts:SetContext"]
     }
  ]
}
```

Además, el rol de ejecución del trabajo necesita permisos para los servicios posteriores AWS, que la ejecución del trabajo invocaría para obtener datos mediante la identidad del usuario. Consulte los siguientes enlaces para configurar S3 Access Grant y Lake Formation.

- Using Lake Formation with EMR Serverless
- Uso de Concesiones de acceso a Amazon S3 con EMR sin servidor

Propagación de identidades de confianza para cargas de trabajo interactivas con EMR sin servidor

Los pasos para propagar identidades a las cargas de trabajo interactivas a través de un punto de conexión de Apache Livy dependen de si los usuarios interactúan con un entorno de desarrollo administrado de AWS como Amazon SageMaker AI o con su propio entorno de Notebook autoalojado como aplicación orientada al cliente.



Entorno de desarrollo administrado por AWS

La siguiente aplicación administrada de AWS orientada al cliente admite la propagación de identidades de confianza con un punto de conexión Apache Livy de EMR sin servidor:

- Amazon SageMaker AI

Entorno de cuaderno autoalojado administrado por el cliente

Para permitir la propagación de identidades de confianza para los usuarios de aplicaciones desarrolladas a medida, consulte Access AWS services programmatically using trusted identity propagation en el Blog de seguridad de AWS.

Consideraciones y limitaciones de la integración de EMR sin servidor para la propagación de identidades de confianza

Tenga en cuenta estos puntos cuando utilice la propagación de identidades de confianza de IAM Identity Center con la aplicación de EMR sin servidor:

- La propagación de identidades de confianza a través de Identity Center es compatible con Amazon EMR 7.8.0 y versiones posteriores, y solo con Apache Spark.
- La propagación de identidades de confianza solo se puede utilizar para cargas de trabajo interactivas con EMR sin servidor a través de un punto de conexión Apache Livy. Las cargas de trabajo interactivas a través de EMR Studio no admiten la propagación de identidades de confianza
- Los trabajos por lotes y de transmisión no admiten la propagación de identidades de confianza

- Los controles de acceso detallados que utilizan AWS Lake Formation y cuentan con propagación de identidades de confianza están disponibles para cargas de trabajo interactivas con EMR sin servidor a través de un punto de conexión de Apache Livy.
- La propagación de identidades de confianza con Amazon EMR está disponible en las siguientes regiones de AWS:
  - af-south-1: África (Ciudad del Cabo)
  - ap-east-1: Asia-Pacífico (Hong Kong)
  - ap-northeast-1 – Asia-Pacífico (Tokio)
  - ap-northeast-2: Asia Pacífico (Seúl)
  - ap-northeast-3: Asia-Pacífico (Osaka)
  - ap-south-1: Asia Pacífico (Mumbai)
  - ap-southeast-1 – Asia-Pacífico (Singapur)
  - ap-southeast-2 – Asia-Pacífico (Sídney)
  - ap-southeast-3: Asia-Pacífico (Yakarta)
  - ca-central-1: Canadá (Central)
  - ca-west-1: Canadá (Calgary)
  - eu-central-1 – Europa (Fráncfort)
  - eu-north-1: Europa (Estocolmo)
  - eu-south-1: Europa (Milán)
  - eu-south-2: Europa (España)
  - eu-west-1 – Europa (Irlanda)
  - eu-west-2: Europa (Londres)
  - eu-west-3: Europa (París)
  - me-central-1: Medio Oriente (EAU)
  - me-south-1: Medio Oriente (Baréin)
  - sa-east-1: América del Sur (São Paulo)
  - us-east-1 – EE. UU. Este (Norte de Virginia)
  - us-east-2 – EE. UU. Este (Ohio)
  - us-west-1: EE. UU. Oeste (Norte de California)
  - us-west-2 – EE. UU. Oeste (Oregón)

Uso de Lake Formation con EMR sin servidor

Aprenda a utilizar Lake Formation para establecer un control de acceso detallado.

Acceso sin filtro a Lake Formation para EMR sin servidor

Con Amazon EMR 7.8.0 y versiones posteriores, puede aprovechar AWS Lake Formation con Catálogo de datos de Glue, donde el rol de tiempo de ejecución del trabajo cuenta con permisos de tabla completos sin las limitaciones del control de acceso detallado. Esta capacidad le permite leer y escribir en tablas protegidas por Lake Formation desde sus trabajos de Spark interactivos y por lotes de EMR sin servidor. Consulte las siguientes secciones para obtener más información sobre Lake Formation y cómo usarlo con EMR sin servidor.

Uso de Lake Formation con acceso completo a las tablas

Puede acceder a las tablas de Catálogo de datos de Glue protegidas por AWS Lake Formation desde los trabajos de Spark de EMR sin servidor o desde sesiones interactivas en las que el rol de tiempo de ejecución del trabajo tiene acceso completo a las tablas. No necesita habilitar AWS Lake Formation en la aplicación de EMR sin servidor. Cuando un trabajo de Spark en se configura para acceso completo a las tablas (FTA), se utilizan las credenciales de AWS Lake Formation para leer o escribir datos en S3 de las tablas registradas en AWS Lake Formation, mientras que se utilizan las credenciales del rol de tiempo de ejecución del trabajo para acceder a tablas que no estén registradas en AWS Lake Formation.

> **Important**
>
> No habilite AWS Lake Formation para el control de acceso detallado. Un trabajo no puede utilizar simultáneamente el acceso completo a las tablas (FTA) y el control de acceso detallado (FGAC) en el mismo clúster o la misma aplicación de EMR.

Paso 1: habilitación del acceso completo a las tablas en Lake Formation

Para utilizar el modo de acceso completo a la tabla (FTA), debe permitir que motores de consulta de terceros accedan a los datos sin la validación de etiquetas de sesión de IAM en AWS Lake Formation. Para habilitarlo, siga los pasos descritos en Integración de aplicaciones para acceso completo a la tabla .

> **Note**
>
> Cuando se accede a tablas entre cuentas, se debe habilitar el acceso completo a las tablas tanto en las cuentas de productores como en las de consumidores. Del mismo modo, cuando se accede a las tablas entre regiones, esta configuración debe estar habilitada tanto en las regiones de productores como en las de consumidores.

Paso 2: configuración de los permisos de IAM para el rol en tiempo de ejecución del trabajo

Para acceder a los datos subyacentes, ya sea para lectura o escritura, además de los permisos de Lake Formation, el rol de tiempo de ejecución del trabajo necesita el permiso de IAM `lakeformation:GetDataAccess`. Con este permiso, Lake Formation concede la solicitud de credenciales temporales para acceder a los datos.

El siguiente es un ejemplo de política sobre cómo proporcionar permisos de IAM para acceder a un script en Amazon S3, cargar registros en S3, permisos de la API de AWS Glue y permiso para acceder a Lake Formation.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
```

```
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Paso 2.1: configuración de los permisos de Lake Formation

- Los trabajos de Spark que leen datos desde S3 requieren el permiso SELECCIONAR de Lake Formation.
- Los trabajos de Spark que escriben o eliminan datos en S3 requieren el permiso TODOS (SUPER) de Lake Formation.
- Los trabajos de Spark que interactúan con el Catálogo de datos de Glue requieren los permisos DESCRIBIR, MODIFICAR o DESCARTAR, según corresponda.

Para obtener más información, consulte Granting permissions on Data Catalog resources.

Paso 3: inicialización de una sesión de Spark para el acceso completo a la tabla mediante Lake Formation

Requisitos previos

El Catálogo de datos de AWS Glue se debe configurar como un repositorio de metadatos para acceder a las tablas de Lake Formation.

Para configurar el catálogo de Glue como repositorio de metadatos, establezca la siguiente configuración:

```
--conf spark.sql.catalogImplementation=hive
--conf
 spark.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCata
```

Para obtener más información sobre cómo habilitar el Catálogo de datos para EMR sin servidor, consulte Metastore configuration for EMR Serverless.

Para acceder a tablas registradas en AWS Lake Formation, es necesario establecer las siguientes configuraciones durante la inicialización de Spark, con el fin de configurar Spark para que utilice credenciales de AWS Lake Formation.

Hive

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=S3_DATA_LOCATION
--conf spark.sql.catalog.spark_catalog.client.region=REGION
--conf spark.sql.catalog.spark_catalog.type=glue
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Delta Lake

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.A` configure el sistema de archivos EMR (EMRFS) o EMR S3A para utilizar las credenciales de S3 para acceder a las tablas registradas en AWS Lake Formation. Si la tabla no está registrada, utilice las credenciales del rol en tiempo de ejecución del trabajo.
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` y `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true`: configure EMRFS para que utilice el encabezado de tipo de contenido application/x-directory en lugar del sufijo $folder$ al crear carpetas en S3. Esto es necesario para leer las tablas de Lake Formation, ya que las credenciales de Lake Formation no permiten leer carpetas de tablas que utilicen el sufijo $folder$.
- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true`: configure Spark para omitir la validación de que la ubicación de la tabla esté vacía antes de crearla. Esto es necesario para las tablas registradas en Lake Formation, ya que las credenciales de Lake Formation necesarias para verificar que la ubicación esté vacía solo se encuentran disponibles después de crear la tabla de Catálogo de datos de Glue. Sin esta configuración, las credenciales del rol en tiempo de ejecución del trabajo validarán si la ubicación de la tabla está vacía.
- `spark.sql.catalog.createDirectoryAfterTable.enabled=true`: configure Spark para crear la carpeta en Amazon S3 después de crear la tabla en el repositorio de metadatos de Hive. Esto es necesario para tablas registradas en Lake Formation, ya que las credenciales de Lake Formation necesarias para crear la carpeta en S3 solo se encuentran disponibles después de crear la tabla de Catálogo de datos de Glue.
- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true`: configure Spark para descartar la carpeta en S3 antes de eliminar la tabla del repositorio de metadatos de Hive. Esto es necesario para las tablas registradas en Lake Formation, ya que las credenciales de Lake

Formation para descartar la carpeta de S3 no están disponibles después de eliminar la tabla del Catálogo de datos de Glue.

- `spark.sql.catalog.<catalog>.glue.lakeformation-enabled=true`: configure el catálogo de Iceberg para que utilice las credenciales de AWS Lake Formation de S3 para acceder a tablas registradas en Lake Formation. Si la tabla no está registrada, utilice las credenciales predeterminadas del entorno.

Configuración del modo de acceso completo a las tablas en Sagemaker Unified Studio

Para acceder a tablas las registradas en Lake Formation desde sesiones interactivas de Spark en blocs de notas de JupyterLab, debe utilizar el modo de permisos de compatibilidad. Use el comando mágico %%configure para configurar Spark. Elija la configuración según el tipo de tabla:

For Hive tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

For Iceberg tables

```
%%configure -f
{
    "conf": {
        "spark.sql.catalog.spark_catalog":
 "org.apache.iceberg.spark.SparkSessionCatalog",
        "spark.sql.catalog.spark_catalog.warehouse": "S3_DATA_LOCATION",
        "spark.sql.catalog.spark_catalog.client.region": "REGION",
        "spark.sql.catalog.spark_catalog.type": "glue",
        "spark.sql.catalog.spark_catalog.glue.account-id": "ACCOUNT_ID",
        "spark.sql.catalog.spark_catalog.glue.lakeformation-enabled": "true",
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": "true",
```

```
    }
}
```

For Delta Lake tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

Reemplace los marcadores de posición:

- `S3_DATA_LOCATION`: la ruta de su bucket de S3
- `REGION`: región AWS (p. ej., us-east-1)
- `ACCOUNT_ID`: su ID de cuenta de AWS

> **Note**
>
> Debe establecer estas configuraciones antes de ejecutar cualquier operación de Spark en el bloc de notas.

Operaciones admitidas

Estas operaciones usarán credenciales de AWS Lake Formation para acceder a los datos de las tablas.

- CREATE TABLE
- ALTER TABLE
- INSERT INTO

- INSERT OVERWRITE
- UPDATE
- MERGE INTO
- DELETE FROM
- ANALIZAR TABLA
- REPARAR TABLA
- DROP TABLE
- Consultas de orígenes de datos de Spark
- Escrituras de orígenes de datos de Spark

> **Note**
>
> Las operaciones no mencionadas anteriormente aún usan permisos de IAM para acceder a los datos de las tablas.

Consideraciones

- Si una tabla de Hive se crea mediante un trabajo que no tiene habilitado el acceso completo a la tabla y no se insertan registros, las lecturas o escrituras posteriores desde un trabajo con acceso completo a la tabla fallarán. Esto se debe a que, cuando no tiene habilitado el acceso completo a la tabla, EMR Spark agrega el sufijo `$folder$` al nombre de la carpeta de la tabla. Para resolver esto, puede optar por una de las siguientes acciones:
  - Insertar al menos una fila en la tabla desde un trabajo que no tenga FTA habilitado.
  - Configure el trabajo que no tiene FTA habilitado para evitar el uso del sufijo `$folder$` en el nombre de la carpeta en S3. Esto se logra al configurar en Spark `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true`.
  - Cree una carpeta de S3 en la ubicación de la tabla `s3://path/to/table/table_name` mediante la consola de AWS S3 o la CLI de AWS S3.
- El acceso completo a las tablas es compatible con el sistema de archivos EMR (EMRFS) a partir de la versión 7.8.0 de Amazon EMR y con el sistema de archivos S3A a partir de la versión 7.10.0 de Amazon EMR.
- El acceso completo a tablas es compatible con tablas de Hive, Iceberg y Delta. Las tablas de Hudi son compatibles con el acceso completo a las tablas.

- Los trabajos que hagan referencia a tablas con reglas de control de acceso detallado (FGAC) de Lake Formation o con vistas del Catálogo de datos de Glue generarán errores. Para consultar una tabla con reglas de FGAC o una vista del Catálogo de datos de Glue, debe usar el modo FGAC. Para habilitar el modo FGAC, siga los pasos que se indican en la documentación de AWS: Using EMR Serverless with AWS Lake Formation for fine-grained access control.
- El acceso completo a tablas no ofrece compatibilidad con Spark Streaming.
- Si se ingresa un marco de datos de Spark en una tabla de Lake Formation, solo se admite el modo ANEXAR para las tablas de Hive e Iceberg: `df.write.mode("append").saveAsTable(table_name)`
- La creación de tablas externas requiere permisos de IAM.
- Como Lake Formation almacena temporalmente en caché las credenciales de un trabajo de Spark, es posible que los trabajos por lotes o las sesiones interactivas de Spark que se estén ejecutando no reflejen los cambios de permisos.
- Debe utilizar un rol definido por el usuario, no uno vinculado a un servicio: Requisitos de Lake Formation en relación con los roles.

Uso de EMR sin servidor con AWS Lake Formation para control de acceso detallado

Descripción general

Con Amazon EMR 7.2.0 y versiones posteriores, puede aprovechar AWS Lake Formation para aplicar controles de acceso detallados en las tablas del Catálogo de datos respaldadas por S3. Esta capacidad le permite configurar controles de acceso a nivel de tabla, fila, columna y celda para consultas deread dentro de sus trabajos de Spark de Amazon EMR sin servidor. Para configurar un control de acceso detallado para los trabajos por lotes y las sesiones interactivas de Apache Spark, utilice EMR Studio. Consulte las siguientes secciones para obtener más información sobre Lake Formation y cómo usarlo con EMR sin servidor.

El uso de Amazon EMR sin servidor con AWS Lake Formation implica cargos adicionales. Para obtener más información, consulte Precios de Amazon EMR.

Cómo funciona EMR sin servidor con AWS Lake Formation

El uso de EMR sin servidor con Lake Formation le permite implementar una capa de permisos en cada trabajo de Spark para aplicar el control de permisos de Lake Formation cuando EMR

sin servidor ejecuta trabajos. EMR sin servidor utiliza 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 implementa las políticas de Lake Formation. Para obtener más información, consulte Qué es AWS Lake Formation y Consideraciones y limitaciones.

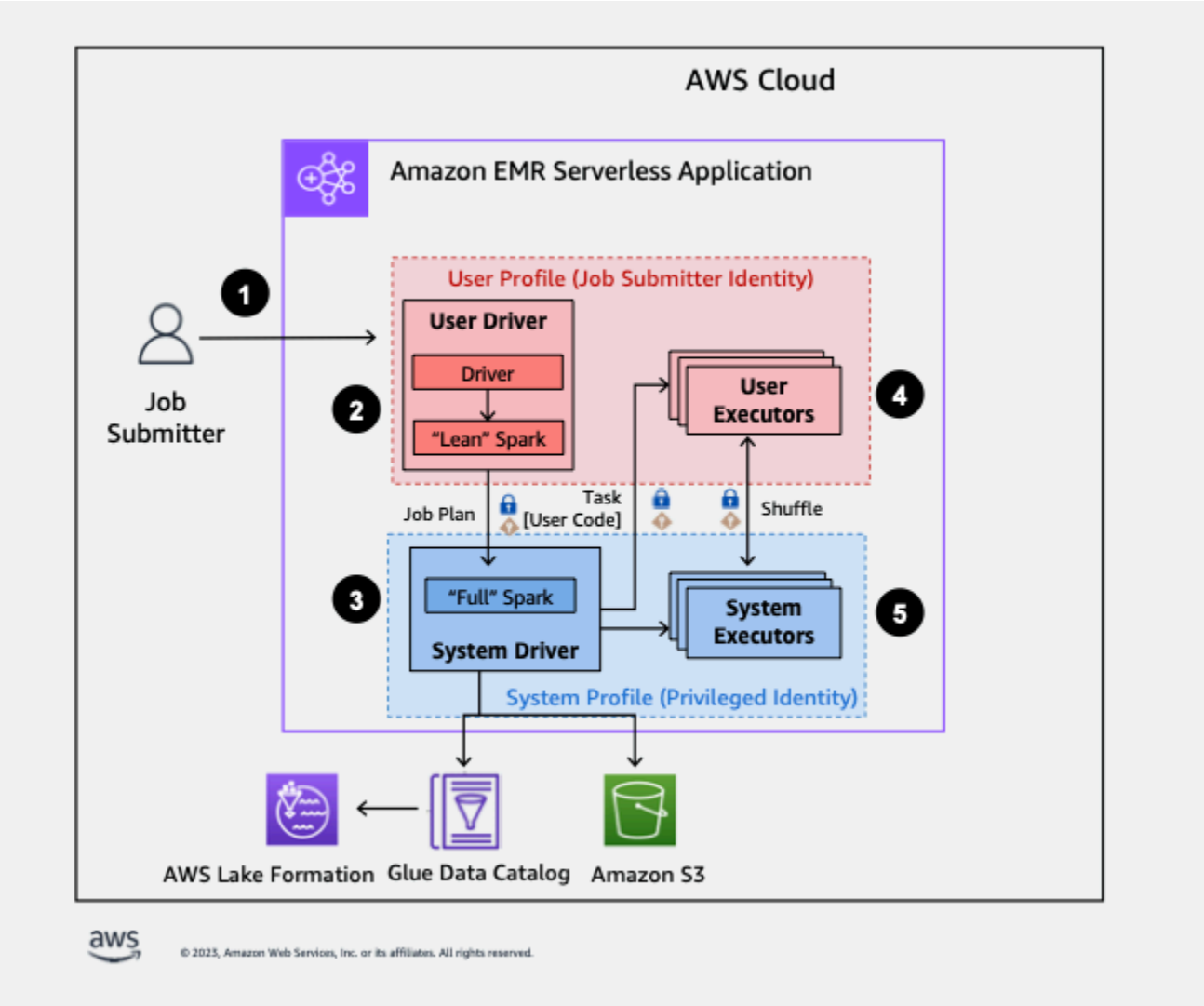
Cuando utilice la capacidad preinicializada con Lake Formation, le sugerimos que tenga un mínimo de dos controladores de Spark. Cada trabajo habilitado para Lake Formation utiliza dos controladores Spark, uno para el perfil de usuario y otro para el perfil del sistema. Para obtener el mejor rendimiento, utilice el doble de controladores para los trabajos habilitados para Lake Formation en comparación con los que utilizaría si no usara Lake Formation.

Cuando ejecute trabajos de Spark en EMR sin servidor, también debe tener en cuenta el impacto de la asignación dinámica en la administración de recursos y el rendimiento del clúster. La configuración de `spark.dynamicAllocation.maxExecutors` del número máximo de ejecutores por perfil de recursos se aplica a los ejecutores de usuario y a los ejecutores del sistema. Si configura ese número para que sea igual al número máximo permitido de ejecutores, es posible que la ejecución del trabajo se bloquee debido a que un tipo de ejecutor utiliza todos los recursos disponibles, lo que impide que el otro ejecutor ejecute los trabajos.

Para no quedarse sin recursos, EMR sin servidor establece el número máximo predeterminado de ejecutores por perfil de recursos en el 90 % del valor `spark.dynamicAllocation.maxExecutors`. Puede anular esta configuración al especificar `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`

A continuación, se ofrece una descripción general de alto nivel sobre cómo EMR sin servidor obtiene acceso a los datos protegidos por las políticas de seguridad de Lake Formation.



1. Un usuario envía un trabajo de Spark a una aplicación EMR sin servidor habilitada con AWS Lake Formation.
2. EMR sin servidor 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. EMR sin servidor configura un segundo controlador denominado controlador del sistema y lo ejecuta en el perfil del sistema (con una identidad privilegiada). EMR sin servidor configura un canal TLS 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 plan de trabajo en una secuencia de etapas de ejecución.

4. Luego, EMR sin servidor 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. Las etapas que leen datos de las tablas del catálogo de datos protegidas por AWS Lake Formation o aquellas que aplican filtros de seguridad se delegan a los ejecutores del sistema.

Habilitación de Lake Formation en Amazon EMR

Para habilitar Lake Formation, debe establecer `spark.emr-serverless.lakeformation.enabled` en `true` según la clasificación de `spark-defaults` para el parámetro de configuración de tiempo de ejecución al crear una aplicación EMR sin servidor.

```
aws emr-serverless create-application \
    --release-label emr-7.10.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

También puede habilitar Lake Formation al crear una nueva aplicación en EMR Studio. Elija Uso de 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 sin servidor, por lo que no necesita volver a habilitar explícitamente el cifrado entre trabajadores.

Habilitación de Lake Formation para trabajos de Spark

Para habilitar Lake Formation para trabajos individuales de Spark, establezca `spark.emr-serverless.lakeformation.enabled` en “verdadero” cuando utilice `spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

Permisos de IAM del rol de tiempo de ejecución de trabajos

Los permisos de Lake Formation controlan el acceso a los recursos de Catálogo de datos de AWS Glue, a las ubicaciones de Amazon S3 y a los datos subyacentes en esas ubicaciones. Los permisos

de IAM controlan el acceso a las API y los recursos de Lake Formation y AWS Glue. 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 permiso de IAM en la operativa de la API `glue:Get*`.

El siguiente es un ejemplo de política sobre cómo proporcionar permisos de IAM para acceder a un script en S3, cargar registros en S3, permisos de la API de Glue de AWS y permiso para acceder a Lake Formation.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts",
        "arn:aws:s3:::*.amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
```

```
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Configuración de permisos de Lake Formation para roles de tiempo de ejecución de trabajos

Primero, registre la ubicación de su tabla de Hive con Lake Formation. A continuación, cree los permisos para el rol de tiempo de ejecución de su trabajo en la tabla que desee. Para obtener más detalles acerca de Lake Formation, consulte Qué es AWS Lake Formation en la Guía para desarrolladores de AWS Lake Formation.

Después de configurar los permisos de Lake Formation, envíe trabajos de Spark en Amazon EMR sin servidor. Para obtener más información acerca de trabajos de Spark, consulte ejemplos de Spark.

Envío de la ejecución de un trabajo

Cuando termine de configurar los permisos de Lake Formation, podrá enviar trabajos a Spark en EMR sin servidor. En la siguiente sección, se muestran ejemplos de cómo configurar y enviar las propiedades de la ejecución del trabajo.

Compatibilidad con el formato de tabla abierta

EMR sin servidor es compatible con Apache Hive, con Apache Iceberg y, a partir de la versión 7.6.0, con Delta Lake y con Apache Hudi. Para obtener más información sobre la compatibilidad del de las operaciones, consulte las siguientes pestañas.

Hive

Operaciones	Notas
Operaciones de lectura	Totalmente compatible
Consultas incrementales	No aplicable
Consultas de viaje en el tiempo	No es aplicable a este formato de tabla
`DML INSERT`	Solo con permisos de IAM
ACTUALIZACIÓN DE DML	No es aplicable a este formato de tabla
`DML DELETE`	No es aplicable a este formato de tabla
Comandos DDL	Solo con permisos de IAM
Tabla de metadatos	No es aplicable a este formato de tabla
Procedimientos almacenados	No aplicable
Características de mantenimiento y utilidad de la tabla	No aplicable

Iceberg

Operaciones	Notas
Operaciones de lectura	Totalmente compatible
Consultas incrementales	Totalmente compatible
Consultas de viaje en el tiempo	Totalmente compatible
`DML INSERT`	Solo con permisos de IAM
ACTUALIZACIÓN DE DML	Solo con permisos de IAM
`DML DELETE`	Solo con permisos de IAM

Operaciones	Notas
Comandos DDL	Solo con permisos de IAM
Tabla de metadatos	Es compatible, pero algunas tablas están ocultas. Consulte consideraciones y limitaciones para obtener más información.
Procedimientos almacenados	Se admite con las excepciones de `register_table` y `migrate`. Consulte consideraciones y limitaciones para obtener más información.
Características de mantenimiento y utilidad de la tabla	No aplicable

Configuración de Spark para Iceberg: en el siguiente ejemplo, se muestra cómo configurar Spark con Iceberg. Para ejecutar trabajos de Iceberg, debe proporcionar las siguientes propiedades `spark-submit`.

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

Hudi

Operaciones	Notas
Operaciones de lectura	Totalmente compatible
Consultas incrementales	Totalmente compatible
Consultas de viaje en el tiempo	Totalmente compatible
`DML INSERT`	Solo con permisos de IAM

Operaciones	Notas
ACTUALIZACIÓN DE DML	Solo con permisos de IAM
DML DELETE	Solo con permisos de IAM
Comandos DDL	Solo con permisos de IAM
Tabla de metadatos	No admitido
Procedimientos almacenados	No aplicable
Características de mantenimiento y utilidad de la tabla	No admitido

En los siguientes ejemplos, se configura Spark con Hudi y se especifican las ubicaciones de los archivos y otras propiedades necesarias para su uso.

Configuración de Spark para Hudi: cuando se usa en un cuaderno, este fragmento especifica la ruta al archivo JAR de la agrupación de Hudi Spark que habilita la funcionalidad de Hudi en Spark. También configura Spark para utilizar el Catálogo de datos de AWS Glue como repositorio de metadatos.

```
%%configure -f
{
    "conf": {
        "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
        "spark.hadoop.hive.metastore.client.factory.class":
 "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
        "spark.serializer": "org.apache.spark.serializer.JavaSerializer",
        "spark.sql.catalog.spark_catalog":
 "org.apache.spark.sql.hudi.catalog.HoodieCatalog",
        "spark.sql.extensions":
 "org.apache.spark.sql.hudi.HoodieSparkSessionExtension"
    }
}
```

Configuración de Spark para Hudi con AWS Glue: cuando se usa en un cuaderno, este fragmento habilita a Hudi como un formato de lago de datos compatible y asegura que las bibliotecas y dependencias de Hudi estén disponibles.

```
%%configure
{
    "--conf": "spark.serializer=org.apache.spark.serializer.JavaSerializer --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog --
conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
    "--datalake-formats": "hudi",
    "--enable-glue-datacatalog": True,
    "--enable-lakeformation-fine-grained-access": "true"
}
```

Delta Lake

Operaciones	Notas
Operaciones de lectura	Totalmente compatible
Consultas incrementales	Totalmente compatible
Consultas de viaje en el tiempo	Totalmente compatible
`DML INSERT`	Solo con permisos de IAM
ACTUALIZACIÓN DE DML	Solo con permisos de IAM
`DML DELETE`	Solo con permisos de IAM
Comandos DDL	Solo con permisos de IAM
Tabla de metadatos	No admitido
Procedimientos almacenados	No aplicable
Características de mantenimiento y utilidad de la tabla	No admitido

EMR sin servidor con Delta Lake: para utilizar Delta Lake con Lake Formation en EMR sin servidor, ejecute el siguiente comando:

```
spark-sql \
```

```
 --conf
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserver
\
 --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \
```

Trabajos de depuración

> **Note**
>
> Con esta característica, puede acceder a los registros stdout y stderr de los trabajadores del perfil del sistema que contengan información confidencial sin filtrar. El siguiente permiso solo se debe usar para acceder a datos que no sean de producción. En el caso de las aplicaciones creadas para uso en trabajos de producción, le recomendamos encarecidamente que incluya estos permisos únicamente para los administradores o usuarios con acceso elevado a los datos.

Con EMR-7.3.0 y versiones posteriores, EMR sin servidor permite la autodepuración para los trabajos por lotes habilitados para Lake Formation. Para ello, utilice el nuevo parámetro accessSystemProfileLogs en la API GetDashboardForJobRun. Si accessSystemProfileLogs está establecido en true, puede acceder a los registros stdout y stderr de los trabajadores del perfil del sistema, que se pueden utilizar para depurar un trabajo por lotes de EMR sin servidor habilitado para Lake Formation.

```
aws emr-serverless get-dashboard-for-job-run \
  --application-id application-id
  --job-run-id job-run-id
  --access-system-profile-logs
```

Permisos necesarios

La entidad principal que quiera depurar trabajos por lotes habilitados para Lake Formation mediante GetDashboardForJobrun debe contar con los siguientes permisos adicionales:

```
{
    "Sid": "AccessSystemProfileLogs",
    "Effect": "Allow",
```

```
    "Action": [
        "emr-serverless:GetDashboardForJobRun",
        "emr-serverless:AccessSystemProfileLogs",
        "glue:GetDatabases",
        "glue:SearchTables"
    ],
    "Resource": [
        "arn:aws:emr-serverless:region:account-id:/applications/applicationId/
jobruns/jobid",
        "arn:aws:glue:region:account-id:catalog",
        "arn:aws:glue:region:account-id:database/*",
        "arn:aws:glue:region:account-id:table/*/*"
    ]
}
```

Consideraciones

Los registros del perfil del sistema para la depuración están visibles para los trabajos que accedan a bases de datos o tablas de Lake Formation desde la misma cuenta que el trabajo. Estos no están visibles en los siguientes casos:

- Si el catálogo de datos administrado con los permisos de Lake Formation tiene bases de datos y tablas entre cuentas
- Si el catálogo de datos administrado con los permisos de Lake Formation tiene enlaces a recursos

Trabajo con vistas del Catálogo de datos de Glue

Puede crear y administrar vistas en el Catálogo de datos de AWS Glue para usarlo con EMR sin servidor. Estas se conocen comúnmente como vistas de Catálogo de datos de AWS Glue. Estas vistas son útiles porque admiten varios motores de consultas SQL, lo que permite acceder a la misma vista en diferentes servicios de AWS, como EMR sin servidor, Amazon Athena y Amazon Redshift.

Cuando cree una vista en el Catálogo de datos, utilice concesiones de recursos y controles de acceso basados en etiquetas en AWS Lake Formation para otorgar acceso a dicha vista. Con este método de control de acceso, no necesitas configurar el acceso adicional a las tablas a las que hizo referencia en el momento de crear la vista. Este método de concesión de permisos se denomina semántica del definidor y estas vistas se denominan vistas del definidor. Para obtener más información sobre cómo controlar el acceso en Lake Formation, consulte Granting and revoking permissions on Data Catalog resources en la Guía para desarrolladores de Lake Formation AWS.

Las vistas del Catálogo de datos son útiles para los siguientes casos de uso:

- Control de acceso detallado: puede crear una vista que restrinja el acceso a los datos en función de los permisos que necesite el usuario. Por ejemplo, puede usar las vistas del catálogo de datos para evitar que el personal que no trabaje en el departamento de Recursos Humanos (RR.HH.) vea información de identificación personal (PII).
- Definición de vista completa: aplicar determinados filtros a la vista del Catálogo de datos garantiza que los registros de datos disponibles en una vista del Catálogo de datos estén siempre completos.
- Seguridad mejorada: la definición de consulta utilizada para crear la vista debe estar completa. Esta ventaja significa que las vistas del Catálogo de datos son menos susceptibles a los comandos SQL de actores malintencionados.
- Compartir datos de forma sencilla: comparta los datos con otras cuentas de AWS sin mover los datos. Para obtener más información, consulte Cross-account data sharing in Lake Formation.

Creación de una vista del catálogo de datos

Existen diferentes formas de crear una vista del Catálogo de datos. Estas incluyen el uso de AWS CLI o de Spark SQL. A continuación, se muestran algunos ejemplos.

Using SQL

A continuación, se muestra la sintaxis para crear una vista del Catálogo de datos. Anote el tipo de vista `MULTI DIALECT`. Esto distingue la vista del Catálogo de datos de otras vistas. El predicado `SECURITY` se especifica como `DEFINER`. Esto indica una vista del Catálogo de datos con semántica `DEFINER`.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment], ...) ]
[ COMMENT view_comment ]
[TBLPROPERTIES (property_name = property_value, ... )]
SECURITY DEFINER
AS query;
```

El siguiente es un ejemplo de declaración `CREATE` según la sintaxis:

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER
AS
```

```
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY order_date
```

También puede crear una vista en modo de ejecución de prueba mediante SQL para probar la creación de la vista sin crear realmente el recurso. Si se utiliza esta opción, se realiza una "ejecución de prueba" que valida la entrada y, si la validación se completa correctamente, devuelve el archivo JSON del objeto de la tabla de AWS Glue que representará la vista. En este caso, no se crea la vista real.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
SECURITY DEFINER
[ SHOW VIEW JSON ]
AS view-sql
```

Using the AWS CLI

> **Note**
>
> Cuando se utiliza el comando CLI, el SQL utilizado para crear la vista no se analiza. Esto puede provocar que se cree la vista, pero que las consultas no se realicen de manera correcta. Asegúrese de probar la sintaxis de SQL antes de crear la vista.

Puede utilizar los siguientes comandos CLI para crear una vista:

```
aws glue create-table --cli-input-json '{
  "DatabaseName": "database",
  "TableInput": {
    "Name": "view",
    "StorageDescriptor": {
      "Columns": [
        {
          "Name": "col1",
          "Type": "data-type"
        },
        ...
        {
          "Name": "col_n",
          "Type": "data-type"
        }
```

```
      ],
      "SerdeInfo": {}
    },
    "ViewDefinition": {
      "SubObjects": [
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-table1",
        ...
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-tableN",
       ],
      "IsProtected": true,
      "Representations": [
        {
          "Dialect": "SPARK",
          "DialectVersion": "1.0",
          "ViewOriginalText": "Spark-SQL",
          "ViewExpandedText": "Spark-SQL"
        }
      ]
    }
  }
}'
```

Operaciones de vista admitidas

Los siguientes fragmentos de comandos muestran varias formas de trabajar con las vistas del Catálogo de datos:

- CREATE VIEW

  Crea una vista del Catálogo de datos. El siguiente es un ejemplo que muestra cómo crear una vista a partir de una tabla existente:

  ```
  CREATE PROTECTED MULTI DIALECT VIEW catalog_view
  SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table
  ```

- ALTER VIEW

  Sintaxis disponible:

  - `ALTER VIEW view_name [FORCE] ADD DIALECT AS query`
  - `ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query`
  - `ALTER VIEW view_name DROP DIALECT`

Puede utilizar la opción `FORCE ADD DIALECT` para forzar la actualización del esquema y los subobjetos según el nuevo dialecto del motor. Tenga en cuenta que hacer esto puede provocar errores de consulta si no utiliza también `FORCE` para actualizar otros dialectos del motor. A continuación, se muestra un ejemplo:

```
ALTER VIEW catalog_view FORCE ADD DIALECT
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY orderdate;
```

A continuación, se muestra cómo modificar una vista para actualizar el dialecto:

```
ALTER VIEW catalog_view UPDATE DIALECT AS
SELECT count(*) FROM my_catalog.my_database.source_table;
```

- DESCRIBE VIEW

  Sintaxis disponible para describir una vista:

  - `SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]`: si el usuario tiene los permisos de AWS Glue y Lake Formation necesarios para describir la vista, puede enumerar las columnas. A continuación, se muestran un par de comandos de ejemplo para mostrar columnas:

    ```
    SHOW COLUMNS FROM my_database.source_table;
    SHOW COLUMNS IN my_database.source_table;
    ```

  - `DESCRIBE view_name`: si el usuario tiene los permisos de AWS Glue y Lake Formation necesarios para describir la vista, puede enumerar las columnas de la vista junto con sus metadatos.

- DROP VIEW

  Sintaxis disponible:

  - `DROP VIEW [ IF EXISTS ] view_name`

    El siguiente ejemplo muestra una instrucción `DROP` que comprueba si existe una vista antes de eliminarla:

```
DROP VIEW IF EXISTS catalog_view;
```

- SHOW CREATE VIEW
  - `SHOW CREATE VIEW view_name`: Muestra la instrucción SQL que crea la vista especificada. El siguiente es un ejemplo que muestra cómo crear una vista en el catálogo de datos:

```
SHOW CREATE TABLE my_database.catalog_view;
CREATE PROTECTED MULTI DIALECT VIEW my_catalog.my_database.catalog_view (
  net_profit,
  customer_id,
  item_id,
  sold_date)
TBLPROPERTIES (
  'transient_lastDdlTime' = '1736267222')
SECURITY DEFINER AS SELECT * FROM
my_database.store_sales_partitioned_lf WHERE customer_id IN (SELECT customer_id
 from source_table limit 10)
```

- SHOW VIEWS

  Liste todas las vistas en el catálogo, como vistas regulares, vistas multidialecto (MDV) y MDV sin dialecto de Spark. La sintaxis disponible es la siguiente:

  - `SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]`:

    A continuación, se muestra un comando de ejemplo para mostrar vistas:

```
SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
```

Para obtener más información sobre la creación y configuración de vistas de catálogos de datos, consulte Building AWS Glue Data Catalog views en la Guía para desarrolladores de AWS Lake Formation.

Consulta de la vista del Catálogo de datos

Luego de crear una vista del Catálogo de datos, puede consultarla mediante un trabajo de Spark de Amazon EMR sin servidor que tenga habilitado el control de acceso detallado de AWS Lake Formation. El rol de tiempo de ejecución del trabajo debe contar con el permiso de `SELECT` Lake Formation en la vista del Catálogo de datos. No es necesario conceder acceso a las tablas subyacentes a las que se hace referencia en la vista.

Una vez que haya configurado todo, podrá consultar la vista. Por ejemplo, después de crear una aplicación de EMR sin servidor en EMR Studio, ejecute la siguiente consulta para acceder a una vista.

```
SELECT * from my_database.catalog_view LIMIT 10;
```

Una función útil es `invoker_principal`. Proporciona el identificador único del rol de tiempo de ejecución del trabajo de EMRS. Esto se puede usar para controlar el resultado de la vista según la entidad principal que lo invoca. Se puede usar para agregar una condición a la vista que mejore los resultados de la consulta según el rol de llamada. Para utilizar este rol, el rol de tiempo de ejecución del trabajo debe tener permiso para realizar la acción de IAM `LakeFormation:GetDataLakePrincipal`.

```
select invoker_principal();
```

Puede agregar esta función a una cláusula `WHERE`, por ejemplo, para mejorar los resultados de la consulta.

Consideraciones y limitaciones

Tenga en cuenta lo siguiente cuando cree vistas del Catálogo de datos:

- Solo puede crear vistas del Catálogo de datos con Amazon EMR 7.6 o superior.
- El creador de la vista en el Catálogo de datos debe tener acceso a `SELECT` a las tablas base subyacentes a las que accede la vista. La creación de la vista en el Catálogo de datos falla si una tabla base específica tiene filtros de Lake Formation impuestos al rol del creador.
- Las tablas base no deben tener el permiso de lago de datos `IAMAllowedPrincipals` en Lake Formation. Si está presente, se produce el error Las vistas de múltiples dialectos solo pueden hacer referencia a tablas sin permisos IAMAllowedPrincipals.
- La ubicación de Amazon S3 de la tabla debe estar registrada como ubicación de lago de datos de Lake Formation. Si la tabla no está registrada, se produce el error Las vistas de múltiples dialectos solo pueden hacer referencia a tablas administradas por Lake Formation. Para obtener información sobre cómo registrar ubicaciones de Amazon S3 en Lake Formation, consulte Registering an Amazon S3 location en la Guía para desarrolladores de AWS Lake Formation.
- Solo puede crear vistas del catálogo de datos `PROTECTED`. No se admiten las vistas `UNPROTECTED`.

- No puede hacer referencia a tablas en otra cuenta de AWS en una definición de vista del Catálogo de datos. Tampoco se puede hacer referencia a una tabla de la misma cuenta que esté en una región distinta.
- Para compartir datos entre cuentas o regiones, toda la vista debe compartirse entre cuentas y regiones, mediante los enlaces de recursos de Lake Formation.
- No se admiten las funciones definidas por el usuario (UDF).
- Puede utilizar vistas basadas en tablas de Iceberg. También se admiten los formatos de tabla abierta de Apache Hudi y Delta Lake.
- No puede hacer referencia a otras vistas en las vistas del catálogo de datos.
- Un esquema de vista del Catálogo de datos de AWS Glue siempre se almacena en minúsculas. Por ejemplo, si utiliza una instrucción DDL para crear una vista de Catálogo de datos de Glue con una columna denominada `Castle`, la columna creada en el Catálogo de datos de Glue aparecerá en minúsculas en `castle`. Si luego especifica el nombre de columna en una consulta DML como `Castle` o`CASTLE`, EMR Spark usará minúsculas en el nombre para que ejecute la consulta. Sin embargo, el encabezado de la columna se mostrará de la forma en que lo haya especificado en la consulta.

  Si desea que una consulta tire error si el nombre de una columna especificado en la consulta DML no coincide con el nombre de la columna del Catálogo de datos de Glue, configure `spark.sql.caseSensitive=true`.

Consideraciones y limitaciones

Tenga en cuenta las siguientes consideraciones y limitaciones cuando utilice Lake Formation con EMR sin servidor.

> **Note**
>
> Al habilitar Lake Formation para un trabajo de Spark en EMR sin servidor, el trabajo lanza un controlador de sistema y un controlador de usuario. Si especificó 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 especifique. Si elige la capacidad bajo demanda, EMR sin servidor lanza un controlador de sistema además de un controlador de usuario. Para calcular los costes asociados a su trabajo de EMR sin servidor con Lake Formation, utilice Calculadora de precios de AWS.

Amazon EMR sin servidor con Lake Formation está disponible en todas las regiones compatibles con EMR sin servidor.

- Amazon EMR sin servidor admite un control de acceso detallado a través de Lake Formation para las operaciones de lectura en las tablas de Apache Hive, Apache Iceberg, Delta Lake y Hudi. Los formatos de Apache Hive incluyen Parquet, ORC y xSV.
- Las aplicaciones habilitadas para Lake Formation no admiten el uso de imágenes de EMR sin servidor personalizadas.
- No puede desactivar `DynamicResourceAllocation` para los trabajos de Lake Formation.
- Solo puede utilizar Lake Formation con trabajos de Spark.
- EMR sin servidor con Lake Formation solo admite una única sesión de Spark durante un trabajo.
- EMR sin servidor con Lake Formation solo admite consultas de tablas entre cuentas compartidas a través de enlaces de recursos.
- Lo siguiente no es compatible:
  - Conjuntos de datos distribuidos resilientes (RDD)
  - Streaming de Spark
  - Lectura con permisos concedidos de Lake Formation
  - Control de acceso para columnas anidadas
- EMR sin servidor bloquea aquellas funcionalidades que podrían socavar el aislamiento total del controlador del sistema, incluidas las siguientes:
  - UDT, HiveUDFS y cualquier función definida por el usuario que incluya clases personalizadas
  - Orígenes de datos personalizados
  - Suministro de archivos jar adicionales para la extensión, el conector o el metaalmacén de Spark
  - `ANALYZE TABLE`Comando de la
- Para hacer cumplir los controles de acceso, `EXPLAIN PLAN` y las operaciones de DDL, como `DESCRIBE TABLE`, no exponen información restringida.
- EMR sin servidor 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 permisos elevados, los eventos y registros que genera el controlador del sistema pueden incluir información confidencial. Para evitar que usuarios o códigos no autorizados accedan a esta información confidencial, EMR sin servidor deshabilita el acceso a los registros de los controladores del sistema.

Los registros de los perfiles del sistema siempre se conservan en el almacenamiento administrado; esta es una configuración obligatoria que no se puede deshabilitar. Estos registros se almacenan de forma segura y se cifran mediante una clave de KMS administrada por el cliente o de AWS.

Si su aplicación de EMR sin servidor se encuentra en una subred privada con puntos de conexión de VPC para Amazon S3 y usted adjunta una política de puntos de conexión para controlar el acceso, debe incluir los permisos detallados en Almacenamiento administrado en su política de VPC al punto de conexión de la puerta de enlace de S3 para que sus trabajos puedan enviar datos de registro a Amazon S3 administrado de AWS. Para solucionar solicitudes, póngase en contacto con la asistencia técnica de AWS.

- 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 permiso de IAM para el rol de tiempo de ejecución de trabajos de EMR sin servidor. Si configura incorrectamente el rol registrado con la ubicación de la tabla, los trabajos enviados que usen el rol con permisos de IAM de S3 para la ubicación de la tabla fallarán.
- Para escribir en una tabla de Lake Formation se utiliza el permiso de IAM en lugar de los permisos concedidos por Lake Formation. Si el rol de tiempo de ejecución de su trabajo tiene los permisos de S3 necesarios, puede usarlo para ejecutar operaciones de escritura.
- A partir de Amazon EMR 7.9.0, el FGAC de Spark es compatible con el sistema de archivos S3A cuando se usa con el esquema s3a://.

A continuación, se indican las consideraciones y limitaciones cuando se utiliza 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`, `files`, `manifests` y `refs`. Amazon EMR oculta las columnas que pueden contener datos confidenciales, como `partitions`, `path` y `summaries`. 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 procedimientos `register_table` y `migrate` no son compatibles con ninguna tabla.
- Sugerimos utilizar Iceberg DataFrameWriterV2 en lugar de V1.

Solución de problemas

Consulte las secciones siguientes para obtener soluciones de problemas.

Registro

EMR sin servidor utiliza los perfiles de recursos de Spark para dividir la ejecución de los trabajos. EMR sin servidor utiliza el perfil de usuario para ejecutar el código que proporcionó, mientras que el perfil del sistema implementa las políticas de Lake Formation. Puede acceder a los registros de las tareas ejecutadas con el perfil de usuario.

Para obtener más información sobre la depuración de trabajos habilitados para Lake Formation, consulte Debugging jobs.

IU Live y servidor de historial de Spark

La IU Live y el servidor de historial de Spark contienen todos los eventos de Spark generados a partir del perfil de usuario, así como los eventos redactados generados a partir del controlador del sistema.

Puede 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 IU Live, como el número de registros de salida.

Error de trabajo: permisos de Lake Formation insuficientes

Asegúrese de que su rol de tiempo de ejecución de trabajos tenga los permisos para ejecutar SELECT y DESCRIBE en la tabla a la que está accediendo.

Error de trabajo en la ejecución de RDD

Actualmente, EMR sin servidor no admite operaciones de conjuntos de datos distribuidos resilientes (RDD) en trabajos habilitados para Lake Formation.

Imposible acceder a archivos de datos en Amazon S3

Asegúrese de haber registrado la ubicación del lago de datos en Lake Formation.

Excepción de validación de seguridad

EMR sin servidor detectó un error de validación de seguridad. Póngase en contacto con la asistencia técnica de AWS para obtener ayuda.

Uso compartido del catálogo de datos de Glue y tablas entre cuentas de AWS

Puede compartir bases de datos y tablas entre cuentas y seguir utilizando Lake Formation. Para obtener más información, consulte Cross-account data sharing in Lake Formation y ¿Cómo comparto las bases de datos y tablas del catálogo de datos de AWS Glue entre cuentas mediante AWS Lake Formation?.

Cifrado entre trabajadores

Con las versiones 6.15.0 y posteriores de Amazon EMR, habilite la comunicación con cifrado TLS mutuo entre los trabajadores en sus ejecuciones de trabajo de Spark. Cuando está habilitado, EMR sin servidor genera y distribuye automáticamente un certificado único para cada trabajador aprovisionado en sus ejecuciones de trabajo. Cuando estos trabajadores se comunican para intercambiar mensajes de control o transferir datos aleatorios, establecen una conexión TLS mutua y utilizan los certificados configurados para comprobar la identidad de los demás. Si un trabajador no puede verificar otro certificado, se produce un error en el protocolo de enlace TLS y EMR sin servidor interrumpe la conexión entre ellos.

Si utiliza Lake Formation con EMR sin servidor, el cifrado TLS mutuo está activado de forma predeterminada.

Habilitación del cifrado TLS mutuo en EMR sin servidor

Para habilitar el cifrado TLS mutuo en su aplicación Spark, establézcalo en true `spark.ssl.internode.enabled` al crear la aplicación EMR sin servidor. Si utiliza la AWS consola para crear una aplicación EMR sin servidor, elija Usar configuración personalizada, expanda Configuración de la aplicación e introduzca su `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 cifrado TLS mutuo para ejecutar tareas de Spark individuales, establezca `spark.ssl.internode.enabled` en true cuando utilice `spark-submit`.

```
--conf spark.ssl.internode.enabled=true
```

Secrets Manager para la protección de datos con EMR sin servidor

AWS Secrets Manager es un servicio de almacenamiento secreto que protege las credenciales de la base de datos, las claves de API y otra información secreta. A continuación, en el código puede reemplazar las credenciales codificadas por una llamada a la API de Secrets Manager. Esto ayuda a garantizar la integridad del secreto si alguien examina el código, dado que el secreto no está allí. Para obtener información general, consulte la Guía del usuario de AWS Secrets Manager.

Secrets Manager utiliza las claves de AWS Key Management Service para cifrar los secretos. Para obtener más información, consulte Secret encryption and decryption en la Guía del usuario de AWS Secrets Manager.

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 Rotate AWS Secrets Manager secrets en la Guía del usuario de AWS Secrets Manager.

Amazon EMR sin servidor se integra con AWS Secrets Manager para que pueda almacenar sus datos en Secrets Manager y utilizar el ID del secreto en sus configuraciones.

La forma en la que EMR sin servidor utiliza Secrets

Al almacenar los datos en Secrets Manager y utilizar el ID del secreto en las configuraciones de EMR sin servidor, no se transfieren datos de configuración confidenciales a EMR sin servidor en texto plano ni se exponen a API externas. Si indica que un par de clave-valor contiene un ID de secreto para un secreto que ha almacenado en Secrets Manager, EMR sin servidor recupera el secreto cuando envía los datos de configuración a los trabajadores para ejecutar las tareas.

Para indicar que un par de clave-valor para una configuración contiene una referencia a un secreto almacenado en Secrets Manager, añada la anotación `EMR.secret@` al valor de configuración. Para cualquier propiedad de configuración con anotación de ID secreta, EMR sin servidor llama a Secrets Manager y resuelve el secreto en el momento de la ejecución del trabajo.

Para crear un secreto

Para crear un secreto, siga los pasos de Creación de un secreto de AWS Secrets Manager en Guía del usuario de AWS Secrets Manager. En el Paso 3, debe elegir el campo Texto no cifrado para ingresar su valor confidencial.

Uso del secreto en una clasificación de configuración

En el siguiente ejemplo se muestra cómo proporcionar un secreto en una clasificación de configuración en `StartJobRun`. Si desea configurar las clasificaciones para Secrets Manager a nivel de aplicación, consulte Configuración predeterminada de aplicación para EMR sin servidor.

En los ejemplos, reemplace *`SecretName`* por el nombre del secreto que hay que recuperar. Para obtener más información, consulta Para crear un secreto.

En esta sección

- Especificar referencias secretas: Spark
- Especificar referencias secretas

Especificar 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://amzn-s3-demo-bucket/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-
connector-java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=connection-user-
name
            --conf
 spark.hadoop.javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name
```

```
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

Example — Especifica las referencias secretas en 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",
        }
    }
```

Especificar referencias secretas

Example — Especifica 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://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql",
```

```
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
 javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
 hive.metastore.client.factory.class=org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreCli
                    --hiveconf
 javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
                    --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket"
        }
    }
}'
```

Example — Especifica las referencias secretas en 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"
    }
}
```

Concesión de acceso a EMR sin servidor para recuperar el secreto

Para permitir que EMR sin servidor recupere el valor secreto de Secrets Manager, agregue la siguiente declaración de política a su secreto al crearlo. Debe crear su secreto con la clave KMS

administrada por el cliente para que EMR sin servidor lea el valor secreto. Para obtener más información, consulte Permissions for the KMS key en la Guía del usuario de AWS Secrets Manager.

En la siguiente política, sustituya *applicationId* por el ID de su solicitud.

Política de recursos para el secreto

Debe incluir los siguientes permisos en la política de recursos de la entrada secreta AWS Secrets Manager para permitir que EMR sin servidor recupere valores secretos. Para garantizar que solo una aplicación específica pueda recuperar este secreto, puede especificar opcionalmente el ID de la aplicación EMR sin servidor como condición en la política.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
*"
        }
      },
      "Sid": "AllowSECRETSMANAGERGetsecretvalue"
    }
  ]
}
```

Cree su secreto con la siguiente política para la clave administrada por el cliente AWS Key Management Service (AWS KMS):

Política para claves AWS KMS administradas por el cliente

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

Rotación del secreto

La rotación consiste en actualizar periódicamente un secreto. Además, puede configurar AWS Secrets Manager para que rote el secreto automáticamente de acuerdo con la programación que especifique. Así, puede reemplazar los secretos a largo plazo por secretos a corto plazo. Esto ayuda a reducir el riesgo de que se comprometa la seguridad. EMR sin servidor recupera el valor del secreto de una configuración anotada cuando el trabajo pasa a un estado en 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 Concesiones de acceso a Amazon S3 con EMR sin servidor

Información general de Concesiones de acceso a S3 para EMR sin servidor

Con las versiones 6.15.0 y posteriores de Amazon EMR, Concesiones de acceso a Amazon S3 proporciona una solución de control de acceso escalable para aumentar el acceso a los datos de Amazon S3 desde EMR sin servidor. Si cuenta con una configuración de permisos compleja o amplia de datos de S3, puede utilizar Concesiones de acceso para escalar los permisos de datos de S3 para usuarios, roles y aplicaciones.

Utilice S3 Access Grants para incrementar el acceso a los datos de Amazon S3, más allá de los permisos que conceden el rol de tiempo de ejecución o los roles de IAM asociados a las identidades con acceso su clúster de Amazon EMR sin servidor.

Para obtener más información, consulte Managing access with S3 Access Grants for Amazon EMR en la Guía de administración de Amazon EMR y Administración del 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 en Amazon S3. Para conocer los pasos para utilizar S3 Access Grants con otras implementaciones de Amazon EMR, consulte la siguiente documentación:

- Uso de S3 Access Grants con Amazon EMR
- Uso de S3 Access Grants con Amazon EMR en EKS

Lanzamiento de una aplicación de EMR sin servidor con Concesiones de acceso a S3

Puede habilitar S3 Access Grants en EMR sin servidor y ejecutar 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. Configure un rol de ejecución de trabajos para su aplicación EMR sin servidor. Incluya los permisos de IAM necesarios para ejecutar los trabajos de Spark y usar Concesiones de acceso a S3, `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 especifica roles de IAM para la ejecución del trabajo que contienen permisos para acceder directamente a S3, los usuarios pueden acceder a los datos que permita el rol incluso si no cuentan con el permiso de Concesiones de acceso a S3.

2. Inicie la aplicación EMR sin servidor con una etiqueta de versión de Amazon EMR de 6.15.0 o superior y la clasificación `spark-defaults`, 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://amzn-s3-demo-destination-bucket1/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
 --conf spark.executor.instances=1"
        }
    }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
          "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
```

```
      }]
}'
```

Consideraciones sobre el uso de S3 Access Grants con EMR sin servidor

Para obtener información importante sobre soporte, compatibilidad y comportamiento al usar Concesiones de acceso a Amazon S3 con EMR sin servidor, consulte S3 Access Grants considerations with Amazon EMR en la Guía de administración de Amazon EMR.

Registro de llamadas a la API de Amazon EMR sin servidor mediante AWS CloudTrail

Amazon EMR sin servidor se integra con AWS CloudTrail, un servicio que proporciona un registro de las medidas adoptadas por un usuario, un rol o un servicio sin servidor AWS. CloudTrail captura todas las llamadas de la API para EMR sin servidor como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de EMR sin servidor y las llamadas desde el código a las operaciones de la API de EMR sin servidor. Si crea un registro de seguimiento, habilite la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, lo que incluye los eventos para EMR sin servidor. Si no configura un registro de seguimiento, aún puede acceder los eventos más recientes en la consola de CloudTrail en Historial de eventos. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a EMR sin servidor, la dirección IP de origen desde la que se realizó la solicitud, quién realizó la solicitud, cuándo se realizó y otros detalles adicionales.

Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.

Información de EMR sin servidor en CloudTrail

CloudTrail se habilita en su Cuenta de AWS cuando se crea la cuenta. Cuando se produce una actividad en EMR sin servidor, esa actividad se registra en un evento de CloudTrail junto con otros eventos de servicio de AWS en Historial de eventos. Puede acceder, buscar y descargar los últimos eventos de la cuenta de Cuenta de AWS. Para obtener más información, consulte Viewing events with CloudTrail Event history.

Para mantener un registro continuo de los eventos en su Cuenta de AWS, incluidos los eventos de EMR sin servidor, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de

AWS. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte lo siguiente:

- Introducción a la creación de registros de seguimiento
- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de registro de CloudTrail de varias regiones y Recibir archivos de registro de CloudTrail de varias cuentas

CloudTrail registra todas las acciones de EMR sin servidor que se documentan en la referencia de API de EMR sin servidor. Por ejemplo, las llamadas a las acciones `CreateApplication`, `StartJobRun` y `CancelJobRun` generan entradas en los archivos de registros de CloudTrail.

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 credenciales de usuario de AWS Identity and Access Management (IAM) o credenciales de usuario raíz.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte CloudTrail userIdentity element.

Descripción de las entradas del archivo de registro de EMR sin servidor

Un registro de seguimiento es una configuración que permite la entrega de eventos como archivos de registros en un bucket de Amazon S3 que especifique. Los archivos de registro de CloudTrail pueden contener una o varias entradas de registro. Un evento representa una solicitud específica realizada desde un origen cualquiera y contiene información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas públicas a la API, por lo que estas no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de registro de CloudTrail que ilustra la acción `CreateApplication`.

```
{
    "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 sin servidor

Los auditores externos evalúan la seguridad y la conformidad de EMR sin servidor como parte de varios programas de conformidad de AWS, incluidos los siguientes:

- Controles del Sistema y Organizaciones (System and Organization Controls, SOC)
- La norma de seguridad de datos del sector de pagos con tarjeta (PCI DSS)
- Programa Federal de Administración de Riesgos y Autorizaciones (FedRAMP) moderado
- Ley de Portabilidad y Responsabilidad de Seguros Médicos de EE. UU (Health Insurance Portability and Accountability Act, HIPAA).

AWS proporciona una lista actualizada frecuentemente de los servicios de AWS adscritos al ámbito de los programas de conformidad en Servicios de AWS en el ámbito del programa de conformidad.

Los informes de auditoría de terceros están disponibles para su descarga mediante AWS Artifact. Para obtener más información, consulte Downloading Reports in AWS Artifact.

Para obtener más información acerca de los programas de conformidad de AWS, consulte Programas de conformidad de AWS.

Su responsabilidad de conformidad al utilizar EMR sin servidor se determina en función de la sensibilidad de los datos, los objetivos de conformidad de su organización, así como de la legislación y los reglamentos aplicables. Si su uso de EMR sin servidor está sujeto a conformidad con normas como HIPAA, PCI o FedRAMP, AWS proporciona recursos de ayuda:

- En las Guías de inicio rápido de seguridad y conformidad, se incluyen consideraciones sobre arquitectura y se ofrecen pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Guías de cumplimiento para clientes de AWS: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas

para garantizar la seguridad de los Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés), el Consejo de Estándares de Seguridad de la Industria de Tarjetas de Pago (PCI, por sus siglas en inglés) y la Organización Internacional de Normalización (ISO, por sus siglas en inglés)).

- 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 Recursos de conformidad es un conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- AWS Security Hub ofrece una vista integral de su estado de seguridad en AWS que lo ayuda a comprobar la conformidad con las normas del sector de seguridad y las prácticas recomendadas.
- AWS Audit Manager: este Servicio de AWS le ayuda a auditar continuamente el uso de AWS con el fin de simplificar la forma en que administra el riesgo y la conformidad con las normativas y los estándares del sector.

Resiliencia de Amazon EMR sin servidor

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. 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, diseñe y use 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 centros de datos únicos o múltiples.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte Infraestructura global de AWS.

Además de la infraestructura global de AWS, Amazon EMR sin servidor ofrece integración con Amazon S3 a través de EMRFS para ayudarle a satisfacer sus necesidades de resistencia de datos y copias de seguridad.

Seguridad de la infraestructura de Amazon EMR sin servidor

Como se trata de un servicio administrado, Amazon EMR se encuentra protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de AWS y sobre cómo AWS protege la infraestructura, consulte Seguridad en la nube de AWS. Para diseñar su entorno de

AWS siguiendo las prácticas recomendadas de seguridad de infraestructura, consulte Protección de la infraestructura en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a la API publicadas en AWS para acceder a Amazon EMR a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Configuración y análisis de vulnerabilidades en Amazon EMR sin servidor

AWS gestiona las tareas de seguridad básicas, como la aplicación de parches en la base de datos y el sistema operativo (SO) de invitado, la configuración del firewall y la recuperación de desastres. Estos procedimientos han sido revisados y certificados por los terceros pertinentes. Para obtener más información, consulte los siguientes recursos:

- Validación de conformidad para Amazon EMR sin servidor
- Modelo de responsabilidad compartida
- Amazon Web Services: Overview of Security Processes

Cuotas y puntos de conexión para EMR Serverless

Puntos de conexión de servicio

Para conectarse mediante programación a un Servicio de AWS, utilice un punto de conexión. Un punto de conexión es la dirección URL del punto de entrada de un servicio web de AWS. Además de los puntos de conexión estándar de AWS, algunos Servicios de AWS ofrecen puntos de conexión FIPS en regiones seleccionadas. En la tabla siguiente se enumeran los puntos de conexión de servicios para EMR sin servidor. Para obtener más información, consulte Servicio de AWS endpoints.

Puntos de conexión de servicio de EMR sin servidor

Nombre de la región	Región	Punto de conexión	Protocolo
Este de EE. UU. (Ohio)	`us-east-2` (limitado a las siguientes zonas de disponibilidad: `use2-az1`, `use2-az2` y `use2-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-az2`, `use1-az4`, `use1-az5` y `use1-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

Nombre de la región	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-Pacífico (Melbourne)	`ap-southeast-4`	`emr-serverless.ap-southeast-4.amazonaws.com`	HTTPS
Asia-Pacífico (Mumbai)	`ap-south-1`	`emr-serverless.ap-south-1.amazonaws.com`	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
Asia-Pacífico (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS
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 y cac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
Oeste de Canadá (Calgary)	ca-west-1	emr-serverless.ca-west-1.amazonaws.com	HTTPS
Europa (Fráncfort)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS
Europa (Zúrich)	eu-central-2	emr-serverless.eu-central-2.amazonaws.com	HTTPS
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

Nombre de la región	Región	Punto de conexión	Protocolo
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
Israel (Tel Aviv)	il-central-1	emr-serverless.il-central-1.amazonaws.com	HTTPS
Medio Oriente (Baréin)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS
Medio Oriente (EAU)	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
China (Pekín)	cn-north-1 (limitado a las siguientes zonas de disponibilidad: cnn1-az1 y cnn1-az2)	emr-serverless.cn-north-1.amazonaws.com.cn	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
AWS GovCloud (Este de EE. UU.)	`us-gov-east-1`	`emr-serverless.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (Oeste de EE. UU.)	`us-gov-west-1`	`emr-serverless.us-gov-west-1.amazonaws.com`	HTTPS

Service Quotas

Las cuotas de servicio, que también se denominan límites, establecen el número máximo de recursos u operaciones de servicio que puede usar en su Cuenta de AWS. EMR sin servidor recopila las métricas de uso de la cuota de servicio cada minuto y las publica en el espacio de nombres de `AWS/Usage`.

> **Note**
>
> Las cuentas nuevas de AWS pueden tener cuotas iniciales más bajas que pueden aumentar con el tiempo. Amazon EMR sin servidor supervisa el uso de la cuenta dentro de cada Región de AWS y luego aumenta automáticamente las cuotas en función de su uso.

En la tabla siguiente se enumeran las cuotas de servicio para EMR sin servidor. Para obtener más información, consulte Servicio de AWS quotas.

Nombre	Límite predeterminado	¿Ajustable?	Descripción
Máximo de vCPU simultáneas por cuenta	16	Sí	La cantidad máxima de vCPU que se pueden ejecutar simultáneamente para

Nombre	Límite predeterminado	¿Ajustable?	Descripción
			la cuenta en la Región de AWS actual.
Número máximo de trabajos en cola por cuenta	2000	Sí	Número máximo de trabajos en cola para la cuenta en la Región de AWS actual.

Límites de la API

A continuación se describen los límites de la API por región para 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.
StartJobRun	1 transacción por segundo. Ráfaga de 25 transacciones por segundo.

Recurso	Cuota predeterminada
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 EMR sin servidor.

- EMR sin servidor está disponible en las siguientes Regiones de AWS:
  - Este de EE. UU. (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-Pacífico (Mumbai)
  - 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)
  - Medio Oriente (EAU)
  - América del Sur (São Paulo)
  - AWS GovCloud (Este de EE. UU.)
  - AWS GovCloud (Oeste de EE. UU.)

Para obtener una lista de puntos de conexión 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 propiedad `executionTimeoutMinutes` de la API `startJobRun` o el SDK de AWS. Puede establecer `executionTimeoutMinutes` en 0 si quiere que su trabajo se ejecute sin que se agote el tiempo de espera. Por ejemplo, si tiene una aplicación de streaming, puede establecer `executionTimeoutMinutes` en 0 para permitir que el trabajo de streaming se ejecute de forma continua.
- La propiedad `billedResourceUtilization` de la API `getJobRun` muestra la vCPU, la memoria y el almacenamiento totales que AWS ha facturado por 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 conectividad de VPC, un trabajo puede acceder a algunos puntos de conexión del Servicio de AWS en la misma Región de AWS. Entre estos servicios se incluyen Amazon S3, AWS Glue, AWS Lake Formation, Registros de Amazon CloudWatch, AWS KMS, AWS Security Token Service, Amazon DynamoDB y AWS Secrets Manager. Puede habilitar la conectividad de VPC para acceder a otros Servicios de AWS a través de AWS PrivateLink, pero no es obligatorio que lo haga. Para acceder a los servicios externos, cree su aplicación con una VPC.
- EMR sin servidor no admite HDFS. Los discos locales de los trabajadores son almacenamiento temporal que EMR sin servidor usa para mezclar y procesar los datos durante las ejecuciones de los trabajos.

Versiones lanzamiento de Amazon EMR sin servidor

Una versión de Amazon EMR es un conjunto de aplicaciones de código abierto del ecosistema de macrodatos. Cada versión incluye aplicaciones, componentes y características de macrodatos que puede seleccionar para que Amazon EMR sin servidor se implemente y configure cuando ejecute el trabajo.

A partir de la versión 6.6.0 y posteriores de Amazon EMR, implemente EMR sin servidor. Esta opción de implementación no está disponible para versiones de lanzamiento anteriores de Amazon EMR. Al enviar el trabajo, especifique una de las siguientes versiones compatibles.

Temas

- EMR Serverless 6.6.0

EMR Serverless 7.10.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.10.0.

Aplicación	Versión
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0.10.2

Notas de la versión 7.10.0 de EMR sin servidor

- Métricas para EMR sin servidor: las métricas de monitoreo se reestructuran para centrarse en las dimensiones `ApplicationName` y `JobName`. Las métricas más antiguas ya no se actualizarán en el futuro. Para obtener más información, consulte Monitoring EMR Serverless applications and jobs.

EMR Serverless 7.9.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.9.0.

Aplicación	Versión
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.8.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.8.0.

Aplicación	Versión
Apache Spark	3.5.4
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.7.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.7.0.

Aplicación	Versión
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.6.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.6.0.

Aplicación	Versión
Apache Spark	3.5.3
Apache Hive	3.1.3

Aplicación	Versión
Apache Tez	0.10.2

EMR Serverless 7.5.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.5.0.

Aplicación	Versión
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.4.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.4.0.

Aplicación	Versión
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.3.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.3.0.

Aplicación	Versión
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Notas de la versión EMR sin servidor 7.3.0

- Simultaneidad de tareas y colas con EMR sin servicio: la simultaneidad de tareas y la creación de colas están habilitadas de forma predeterminada al crear una nueva aplicación EMR sin servidor en Amazon EMR, versión 7.3.0 o superior. Para obtener más información, consulte the section called “Simultaneidad de trabajos y colas”, donde se detalla cómo empezar a utilizar la simultaneidad y las colas, y también se incluye una lista de características a tener en cuenta.

EMR Serverless 7.2.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 7.2.0.

Aplicación	Versión
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Notas de la versión EMR sin servidor 7.2.0

- Lake Formation con EMR sin servidor: ahora se puede utilizar AWS Lake Formation para aplicar controles de acceso detallados en 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 dentro de sus trabajos de Spark de EMR sin servidor. 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 que están disponibles con la versión 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 que están disponibles con la versión EMR Serverless 7.0.0.

Aplicación	Versión
Apache Spark	3.5.0
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 que están disponibles con la versión EMR Serverless 6.15.0.

Aplicación	Versión
Apache Spark	3.4.1
Apache Hive	3.1.3

Aplicación	Versión
Apache Tez	0.10.2

Notas de la versión 6.15.0 de EMR sin servidor

- Compatibilidad con TLS: con las versiones 6.15.0 y posteriores de Amazon EMR sin servidor, puede habilitar la comunicación con cifrado TLS mutuo entre los trabajadores en sus ejecuciones de trabajo de Spark. Cuando está habilitado, EMR sin servidor genera automáticamente un certificado único para cada trabajador que aprovisiona en virtud de las ejecuciones de un trabajo que los trabajadores utilizan durante el protocolo de enlace TLS para autenticarse entre sí y establecer un canal cifrado para procesar los datos de forma segura. Para obtener más información sobre el cifrado TLS mutuo, consulte Inter-worker encryption.

EMR Serverless 6.14.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión 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 que están disponibles con la versión EMR Serverless 6.13.0.

Aplicación	Versión
Apache Spark	3.4.1

Aplicación	Versión
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 que están disponibles con la versión 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 que están disponibles con la versión EMR Serverless 6.11.0.

Aplicación	Versión
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

Notas de la versión 6.11.0 de EMR sin servidor

- Acceda a los recursos de S3 en otras cuentas: con las versiones 6.11.0 y posteriores, puede configurar varios roles de IAM para asumirlos al acceder a los buckets de Amazon S3 en diferentes cuentas AWS desde EMR sin servidor.

EMR Serverless 6.10.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 6.10.0.

Aplicación	Versión
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Notas de la versión 6.10.0 de EMR sin servidor

- Para las aplicaciones EMR sin servidor con la versión 6.10.0 o posteriores, el valor predeterminado de la propiedad `spark.dynamicAllocation.maxExecutors` es `infinity`. Las versiones anteriores son de forma predeterminada `100`. Para obtener más información, consulta Propiedades de trabajo de Spark.

EMR Serverless 6.9.0

En la siguiente tabla se enumeran las versiones de la aplicación que están disponibles con la versión EMR Serverless 6.9.0.

Aplicación	Versión
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

Notas de la versión 6.9.0 de EMR sin servidor

- La integración de Amazon Redshift para Apache Spark se incluye en las versiones 6.9.0 y posteriores de Amazon EMR. 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 EMR sin servidor.

- La versión 6.9.0 de EMR sin servidor agrega compatibilidad con la arquitectura AWS Graviton2 (arm64). Puede usar el parámetro `architecture` para las API `create-application` y `update-application` para elegir la arquitectura arm64. Para obtener más información, consulta Opciones de la arquitectura de Amazon EMR sin servidor.
- Ahora puede exportar, importar, consultar y unir tablas de Amazon DynamoDB directamente desde sus aplicaciones EMR sin servidor Spark y Hive. Para obtener más información, consulta Conexión a DynamoDB con Amazon EMR sin servidor.

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 que están disponibles con la versión 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 que están disponibles con la versión EMR Serverless 6.7.0.

Aplicación	Versión
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

Cambios específicos del motor, mejoras y problemas resueltos

En la tabla siguiente se enumeran las características específicas de motor.

Cambio	Descripción
Característica	El programador de Tez ahora admite la prioridad 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 que están disponibles con la versión EMR Serverless 6.6.0.

Aplicación	Versión
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

Notas de la versión inicial de EMR sin servidor

- EMR sin servidor admite la clasificación `spark-defaults` de configuración de Spark. Esta clasificación cambia los valores en el archivo XML `spark-defaults.conf` de Spark. Las

clasificaciones de configuración le permiten personalizar las aplicaciones. Para obtener más información, consulte Configure applications.

- EMR sin servidor admite las clasificaciones `hive-site`, `tez-site`, `emrfs-site` y `core-site` de configuración de Hive. Esta clasificación puede cambiar los valores del archivo `hive-site.xml` de Hive, el archivo `tez-site.xml` de Tez, la configuración de EMRFS de Amazon EMR o el archivo `core-site.xml` de Hadoop, respectivamente. Las clasificaciones de configuración le permiten personalizar las aplicaciones. Para obtener más información, consulte Configuración de aplicaciones.

Cambios específicos del motor, mejoras y problemas resueltos

- En la siguiente tabla se enumeran las portabilidades con versiones anteriores de Hive y Tez.

  Cambios en Hive y Tez

Cambio	Descripción
Portabilidad con versiones anteriores	TEZ-4430: se corrigió un problema con la propiedad `tez.task.launch.cmd-opts`
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 que se han realizado en la documentación desde la última versión de EMR sin servidor. Para obtener más información sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS.

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 agregarón los nuevos `Sid` `CloudWatchPolicySt atement` y `EC2Policy Statement` a la política AmazonEMRServerles sServiceRolePolicy.	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	Configuración de varios roles de IAM para que los asuma al acceder a los buckets de Amazon S3 en cuentas diferentes de EMR sin servidor (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 predeterminada de aplicación para EMR sin servidor	25 de septiembre de 2023
Actualización de las propiedad es predeterminadas de Hive	Se actualizaron los valores predeterminados de `hive.driver.disk` ,	12 de septiembre de 2023

	`hive.tez.disk.size` y `hive.tez.auto.reducer.parallelism` , así como las propiedades de trabajo de Hive `tez.grouping.min-size` .	
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 rol del SLR de `AmazonEMRServerlessServiceRolePolicy_` para publicar el uso a nivel de cuenta en el espacio de nombres `"AWS/Usage"` .	20 de abril de 2023
Disponibilidad general (GA) de EMR Serverless	Esta es la primera versión pública de EMR sin servidor.	1 de junio de 2022