



Guía de desarrollo de Amazon EMR en EKS

Amazon EMR



Amazon EMR: Guía de desarrollo de Amazon EMR en EKS

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon EMR en EKS?	1
Arquitectura de Amazon EMR en EKS	2
Descripción de conceptos y terminología de Amazon EMR en EKS	3
Espacio de nombres de Kubernetes	3
Clúster virtual	4
Ejecución de trabajo	4
Contenedores de Amazon EMR	4
Qué ocurre cuando envía trabajo a un clúster virtual de Amazon EMR en EKS	5
Introducción a Amazon EMR en EKS	7
Ejecutar una aplicación de Spark	8
Prácticas recomendadas	14
Seguridad	14
Envío de trabajos de PySpark	14
Almacenamiento	14
Integración con metaalmacenes	15
Debugging	15
Solución de problemas de Amazon EMR en EKS	15
Colocación de nodos	15
Rendimiento	15
Optimización de costos	15
¿Usando AWS Outposts	16
Personalización de las imágenes de Docker	17
Cómo personalizar las imágenes de Docker	17
Requisitos previos	18
Paso 1: recuperar una imagen base de Amazon Elastic Container Registry (Amazon ECR)	18
Paso 2: personalizar una imagen base	19
Paso 3: (opcional, pero recomendado) validar una imagen personalizada	20
Paso 4: publicar una imagen personalizada	22
Paso 5: enviar una carga de trabajo de Spark en Amazon EMR mediante una imagen personalizada	23
Personalice las imágenes de Docker para puntos de conexión interactivos	25
Uso de imágenes multiarquitectura	27
Información sobre cómo seleccionar un URI de imagen base	29

Cuentas de registro de Amazon ECR	30
Consideraciones para personalizar imágenes	31
Ejecución de trabajos de Flink	33
Operador de Kubernetes de Flink	33
Configuración	34
Instalación de la implementación del operador de Flink Kubernetes	35
Ejecutar una aplicación de Flink	36
Permisos de rol de seguridad para ejecutar una aplicación Flink	41
Desinstalación del operador	43
Kubernetes nativo de Flink	44
Configuración	44
Introducción	45
Requisitos de seguridad	47
Personalización de las imágenes de Docker para Flink y FluentD	48
Requisitos previos	48
Recupere una imagen base del Amazon Elastic Container Registry	49
Personalice una imagen base	49
Publique una imagen personalizada	50
Envíe una carga de trabajo de Flink	51
Monitorización	51
Amazon Managed Service para Prometheus	52
Uso de la interfaz de usuario de Flink	54
Uso de la configuración de supervisión	55
Cómo apoya Flink la alta disponibilidad y la resiliencia al trabajo	60
Uso de la alta disponibilidad	60
Optimización de los tiempos de reinicio	67
Desactivación rápida	74
Uso del escalador automático	77
Ajuste automático de los parámetros del escalador automático	79
Mantenimiento y solución de problemas para trabajos de Flink en Amazon EMR en EKS	88
Mantenimiento de aplicaciones de Flink	88
Solución de problemas	90
Versiones compatibles	94
Ejecución de trabajos de Spark	95
StartJobRun	95
Configuración	96

Enviar una ejecución de trabajo con StartJobRun	122
Uso de la clasificación de remitentes de trabajos	125
Uso de la clasificación por defecto del contenedor Amazon EMR	130
Operador de Spark	132
Configuración	133
Introducción	134
Escalado automático vertical	138
Desinstalar	143
Uso de la configuración de supervisión para supervisar Spark	144
Seguridad	151
spark-submit	161
Configuración	161
Introducción	162
Seguridad	163
Apache Livy	169
Configuración	169
Introducción	170
Ejecución de una aplicación de Spark	175
Desinstalación	177
Seguridad	178
Propiedades de instalación	188
Solucione los errores comunes de formato de las variables de entorno	194
Administración de las ejecuciones de trabajos	195
Administrar con la CLI	196
Ejecutar scripts de Spark SQL	202
Estados de ejecuciones de trabajos	205
Ver trabajos en la consola	206
Errores comunes de ejecución de tareas	206
Uso de plantillas de trabajos	212
Creación y uso de una plantilla de trabajo para iniciar la ejecución de un trabajo	213
Definición de parámetros de plantilla de trabajo	215
Control del acceso a las plantillas de trabajos	217
Uso de plantillas de pods	218
Escenarios habituales	219
Habilitación de plantillas de pods con Amazon EMR en EKS	221
Campos de plantilla de pod	223

Consideraciones sobre los contenedores asociados	226
Uso de políticas de reintento	228
Establecer una política de reintento	228
Recuperar el estado de la política	230
Supervisar el trabajo	231
Buscar los registros de controladores	232
Uso de la rotación del registro de eventos de Spark	232
Uso de la rotación de los registros de contenedores de Spark	233
Uso del escalado automático vertical	236
Configuración	236
Introducción	239
Configuración	241
Supervisión de las recomendaciones	247
Desinstalación	248
Ejecución de cargas de trabajo interactivas	250
Resumen de los puntos de conexión interactivos	250
Requisitos previos de los puntos de conexión interactivos	253
AWS CLI	253
eksctl	253
Clúster de Amazon EKS	253
Conceder acceso al clúster	254
Active los roles de IAM para cuentas de servicio	254
Crear un rol de ejecución de trabajos de IAM	254
Conceder acceso a los usuarios	254
Registrar el clúster de Amazon EKS con Amazon EMR	255
Controlador del equilibrador de carga	255
Creación de un punto de conexión interactivo	255
Crear un punto de conexión interactivo	256
Especificar parámetros personalizados	256
.....	258
Parámetros del punto de conexión interactivo	258
Configuración de los ajustes de los puntos de conexión interactivos	259
Monitoreo de trabajos de Spark	260
Plantillas de pods personalizadas	261
Implementación de un pod de JEG en un grupo de nodos	262
Opciones de configuración de JEG	266

Modificación de PySpark los parámetros	267
Imagen de kernel personalizada	267
Supervisión de puntos de conexión interactivos	269
Ejemplos	272
Uso de cuadernos Jupyter autoalojados	272
Creación de un grupo de seguridad	273
Crear un punto de conexión interactivo	273
Obtener la URL del servidor de puerta de enlace	274
Obtener el token de autenticación	274
Implementar el cuaderno	275
Limpieza	281
Obtener información sobre los puntos de conexión interactivos con comandos CLI	281
Enumerar los puntos de conexión interactivos	283
Eliminar punto de conexión interactivo	284
Carga de datos	286
Requisitos previos	286
Introducción	286
Superivisión de trabajos	288
Supervisa los trabajos con Amazon CloudWatch Events	288
Automatice Amazon EMR en EKS con eventos CloudWatch	289
Ejemplo: configurar una regla que invoque a Lambda	290
Supervise el módulo de controladores del trabajo con una política de reintentos mediante Amazon Events CloudWatch	291
Administración de clústeres virtuales	292
Crear un clúster virtual	292
Enumerar los clústeres virtuales	294
Describir un clúster virtual	294
Eliminar un clúster virtual	294
Estados del clúster virtual	294
Tutoriales	295
Uso de Delta Lake	295
Uso de Iceberg	296
Configuraciones de sesión de Spark para la integración del catálogo	297
Uso PyFlink	298
Uso de AWS Glue con Flink	299

Uso de Apache Hudi	302
Envío de un trabajo de Apache Hudi	302
Uso de RAPIDS para Spark	306
Uso de Spark en Redshift	310
Lanzar una aplicación de Spark	311
Autenticarse en Amazon Redshift	312
Lectura y escritura en Amazon Redshift	314
Consideraciones	316
Uso de Volcano	317
Descripción general	317
Instalación	318
Enviar: operador de Spark	319
Enviar: spark-submit	321
Usando YuniKorn	322
Descripción general	322
Cree su clúster de	322
Instalar YuniKorn	325
Enviar: operador de Spark	325
Enviar: spark-submit	328
Seguridad	14
Prácticas recomendadas	332
Aplicar el principio de privilegio mínimo	332
Lista de control de acceso para puntos de conexión	332
Obtener las actualizaciones de seguridad más recientes para imágenes personalizadas	333
Limitar el acceso a las credenciales del pod	333
Aislar el código de aplicación no confiable	333
Permisos de control de acceso basado en roles (RBAC)	333
Restringir el acceso a las credenciales del perfil de instancia o rol de IAM del grupo de nodos	334
Protección de los datos	335
Cifrado en reposo	336
Cifrado en tránsito	338
Identity and Access Management	339
Público	339
Autenticación con identidades	340
Administración de acceso mediante políticas	344

Cómo funciona Amazon EMR en EKS con IAM	347
Uso de roles vinculados a servicios	353
Políticas administradas para Amazon EMR en EKS	357
Uso de roles de ejecución de trabajos con Amazon EMR en EKS	358
Ejemplos de políticas basadas en identidades	361
Políticas para el control de acceso basado en etiquetas	364
Solución de problemas	367
Uso de Amazon EMR en EKS con Lake Formation AWS	369
Cómo funciona Amazon EMR en EKS con Lake Formation AWS	370
Habilite Lake Formation con Amazon EMR en EKS	372
Consideraciones y limitaciones	380
Solución de problemas	383
Registro y supervisión	384
Cifrado de registros	385
CloudTrail registros	388
Permisos de acceso de S3	391
Descripción general	391
Lance un clúster	391
Consideraciones	393
Validación de la conformidad	393
Resiliencia	393
Seguridad de infraestructuras	394
Configuración y análisis de vulnerabilidades	394
Puntos de conexión de VPC de la interfaz	395
Crear una política de punto de conexión de VPC para Amazon EMR en EKS	396
Acceso entre cuentas	399
Requisitos previos	399
Cómo acceder a un bucket de Amazon S3 en diversas cuentas o a una tabla de DynamoDB	399
Etiquetado de recursos	404
Conceptos básicos de etiquetas	404
Etiquetar los recursos	405
Restricciones de las etiquetas	406
AWS CLI Trabaje con etiquetas mediante la API Amazon EMR en EKS	407
Solución de problemas	15
Errores en el trabajo de PVC	409

Verificación	409
Parche	410
Parche manual	414
Errores de escalado automático vertical	416
Error 403 Prohibido	416
No se encontró el espacio de nombres	416
Error en las credenciales de Docker	417
Errores del operador de Spark	417
No se pudo instalar el gráfico de Helm	417
Excepción del sistema de archivos no compatible	418
Puntos de conexión de servicio y cuotas	419
Puntos de conexión de servicio	419
Service Quotas	422
Consulte las cuotas y solicite aumentos de cuota	423
Versiones de lanzamiento	424
Versiones 7.9.0	425
Versiones	425
Notas de la versión	427
Cambios	429
emr-7.9.0-latest	429
emr-7.9.0-20250425	429
emr-7.9.0-Flink-última	429
emr-7.9.0-flink-20250425	430
Versiones 7.8.0	430
Versiones	430
Notas de la versión	432
Cambios	433
emr-7.8.0-último	434
emr-7.8.0-20250228	434
emr-7.8.0-flink-latest	434
emr-7.8.0-flink-20250228	434
Versiones 7.7.0	435
Versiones	435
Notas de la versión	437
Cambios	438
emr-7.7.0-latest	438

emr-7.7.0-20250131	439
emr-7.7.0-Flink-latest	439
emr-7.7.0-flink-20250131	439
Versiones 7.6.0	439
Versiones	440
Notas de la versión	441
Características	443
Cambios	443
emr-7.6.0-latest	443
emr-7.6.0-20241213	443
emr-7.6.0-flink-latest	444
emr-7.6.0-flink-20241213	444
Versiones 7.5.0	444
Versiones	444
Notas de la versión	445
Versiones 7.4.0	445
Versiones	445
Notas de la versión	445
Versiones 7.3.0	446
Versiones	446
Notas de la versión	447
Características	449
Cambios	450
emr-7.3.0-latest	450
emr-7.3.0-29240920	450
emr-7.3.0-flink-latest	450
emr-7.3.0-flink-29240920	451
Versiones 7.2.0	451
Versiones	451
Notas de la versión	453
Características	454
emr-7.2.0-latest	455
emr-7.2.0-20240610	455
emr-7.2.0-flink-latest	456
emr-7.2.0-flink-20240610	456
Versiones 7.1.0	456

Versiones	456
Notas de la versión	458
Características	460
emr-7.1.0-latest	460
emr-7.1.0-20240321	460
emr-7.1.0-flink-latest	460
emr-7.1.0-flink-20240321	461
Versiones 7.0.0	461
Versiones	461
Notas de la versión	463
Características	464
Cambios	465
emr-7.0.0-latest	465
emr-7.0.0-2024321	465
emr-7.0.0-20231211	465
emr-7.0.0-flink-latest	466
emr-7.0.0-flink-2024321	466
emr-7.0.0-flink-20231211	466
Versiones de 6.15.0	467
Versiones	467
Notas de la versión	468
Características	470
emr-6.15.0-latest	470
emr-6.15.0-20240105	471
emr-6.15.0-20231109	471
emr-6.15.0-flink-latest	471
emr-6.15.0-flink-20240105	471
emr-6.15.0-flink-20231109	472
Versiones de 6.14.0	472
Versiones	472
Notas de la versión	473
Características	475
emr-6.14.0-latest	475
emr-6.14.0-20231005	475
Versiones de 6.13.0	476
Versiones	476

Notas de la versión	477
Características	479
emr-6.13.0-latest	479
emr-6.13.0-20230814	479
Versiones de 6.12.0	480
Versiones	480
Notas de la versión	481
Características	482
emr-6.12.0-latest	482
emr-6.12.0-20240321	483
emr-6.12.0-20230701	483
Versiones de 6.11.0	483
Versiones	483
Notas de la versión	484
Características	485
emr-6.11.0-latest	486
emr-6.11.0-20230905	486
emr-6.11.0-20230509	487
Versiones de 6.10.0	487
emr-6.10.0-latest	490
emr-6.10.0-20230905	490
emr-6.10.0-20230624	490
emr-6.10.0-20230421	490
emr-6.10.0-20230403	491
emr-6.10.0-20230220	491
Versiones de 6.9.0	491
emr-6.9.0-latest	494
emr-6.9.0-20230905	495
emr-6.9.0-20230624	495
emr-6.9.0-20221108	495
Versiones de 6.8.0	495
emr-6.8.0-latest	500
emr-6.8.0-20230905	500
emr-6.8.0-20230624	500
emr-6.8.0-20221219	501
emr-6.8.0-20220802	501

Versiones de 6.7.0	501
emr-6.7.0-latest	503
emr-6.7.0-20240321	503
emr-6.7.0-20230624	504
emr-6.7.0-20221219	504
emr-6.7.0-20220630	504
Versiones de 6.6.0	504
emr-6.6.0-latest	506
emr-6.6.0-20240321	506
emr-6.6.0-20230624	507
emr-6.6.0-20221219	507
emr-6.6.0-20220411	507
Versiones de 6.5.0	507
emr-6.5.0-latest	509
emr-6.5.0-20240321	509
emr-6.5.0-20221219	509
emr-6.5.0-20220802	510
emr-6.5.0-20211119	510
Versiones de 6.4.0	510
emr-6.4.0-latest	512
emr-6.4.0-20240321	512
emr-6.4.0-20221219	512
emr-6.4.0-20210830	512
Versiones de 6.3.0	513
emr-6.3.0-latest	514
emr-6.3.0-20240321	514
emr-6.3.0-20220802	515
emr-6.3.0-20211008	515
emr-6.3.0-20210802	515
emr-6.3.0-20210429	515
Versiones de 6.2.0	516
emr-6.2.0-latest	517
emr-6.2.0-20240321	517
emr-6.2.0-20220802	518
emr-6.2.0-20211008	518
emr-6.2.0-20210802	518

emr-6.2.0-20210615	519
emr-6.2.0-20210129	519
emr-6.2.0-20201218	519
emr-6.2.0-20201201	519
Versiones de 5.36.0	520
emr-5.36.0-latest	521
emr-5.36.0-20240321	521
emr-5.36.0-20221219	522
emr-5.36.0-20220620	522
emr-5.36.0-20220525	522
Versiones de 5.35.0	522
emr-5.35.0-latest	524
emr-5.35.0-20240321	524
emr-5.35.0-20221219	524
emr-5.35.0-20220802	525
emr-5.35.0-20220307	525
Versiones de 5.34	525
emr-5.34.0-latest	526
emr-5.34.0-20240321	527
emr-5.34.0-20220802	527
emr-5.34.0-20211208	527
Versiones de 5.33.0	527
emr-5.33.0-latest	529
emr-5.33.0-20240321	529
emr-5.33.0-20221219	529
emr-5.33.0-20220802	530
emr-5.33.0-20211008	530
emr-5.33.0-20210802	530
emr-5.33.0-20210615	531
emr-5.33.0-20210323	531
Versiones de 5.32.0	531
emr-5.32.0-latest	532
emr-5.32.0-20240321	533
emr-5.32.0-20220802	533
emr-5.32.0-20211008	533
emr-5.32.0-20210802	534

emr-5.32.0-20210615	534
emr-5.32.0-20210129	534
emr-5.32.0-20201218	534
emr-5.32.0-20201201	535
Historial de documentos	536
.....	dxxxix

¿Qué es Amazon EMR en EKS?

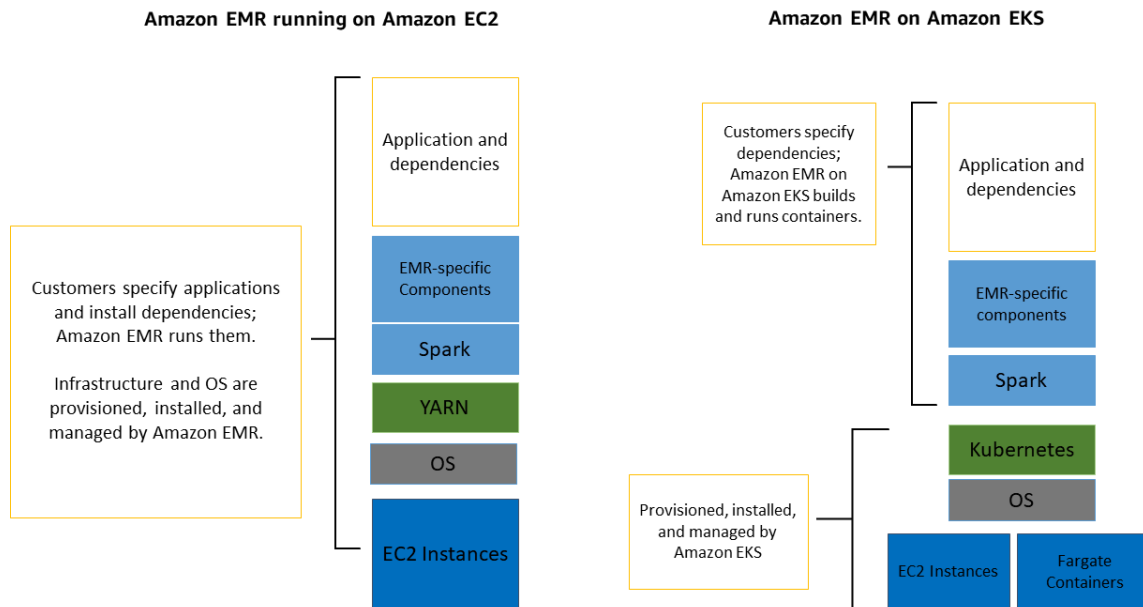
Amazon EMR en EKS proporciona una opción de implementación para Amazon EMR que le permite ejecutar marcos de macrodatos de código abierto en Amazon Elastic Kubernetes Service (Amazon EKS). Con esta opción de implementación, puede centrarse en ejecutar cargas de trabajo de análisis mientras Amazon EMR en EKS crea, configura y administra contenedores para aplicaciones de código abierto.

Si ya utiliza Amazon EMR, ahora puede ejecutar aplicaciones basadas en Amazon EMR con otros tipos de aplicaciones en el mismo clúster de Amazon EKS. Esta opción de implementación también mejora el uso de los recursos y simplifica la administración de la infraestructura en varias zonas de disponibilidad. Si ya ejecuta marcos de macrodatos en Amazon EKS, ahora puede usar Amazon EMR para automatizar el aprovisionamiento y la administración y ejecutar Apache Spark con mayor rapidez.

Amazon EMR en EKS permite a su equipo colaborar de forma más eficiente y procesar grandes cantidades de datos de forma más sencilla y rentable:

- Puede ejecutar aplicaciones en un conjunto común de recursos sin tener que aprovisionar infraestructura. Puede usar [Amazon EMR Studio](#) y el AWS SDK o desarrollar, enviar y diagnosticar aplicaciones de análisis AWS CLI que se ejecuten en clústeres de EKS. Puede ejecutar trabajos programados en Amazon EMR en EKS mediante Apache Airflow autoadministrado o Amazon Managed Workflows para Apache Airflow (MWAA).
- Los equipos de infraestructura pueden administrar de forma centralizada una plataforma informática común para consolidar las cargas de trabajo de Amazon EMR con otras aplicaciones basadas en contenedores. Puede simplificar la administración de la infraestructura con herramientas comunes de Amazon EKS y aprovechar un clúster compartido para cargas de trabajo que necesitan diferentes versiones de marcos de código abierto. También puede reducir la sobrecarga operativa con la administración automatizada de clústeres de Kubernetes y la aplicación de parches al sistema operativo. Con Amazon EC2 y AWS Fargate, puede habilitar varios recursos de cómputo para cumplir con los requisitos de rendimiento, operativos o financieros.

En el siguiente diagrama, se muestran los dos modelos de implementación diferentes de Amazon EMR.



Temas

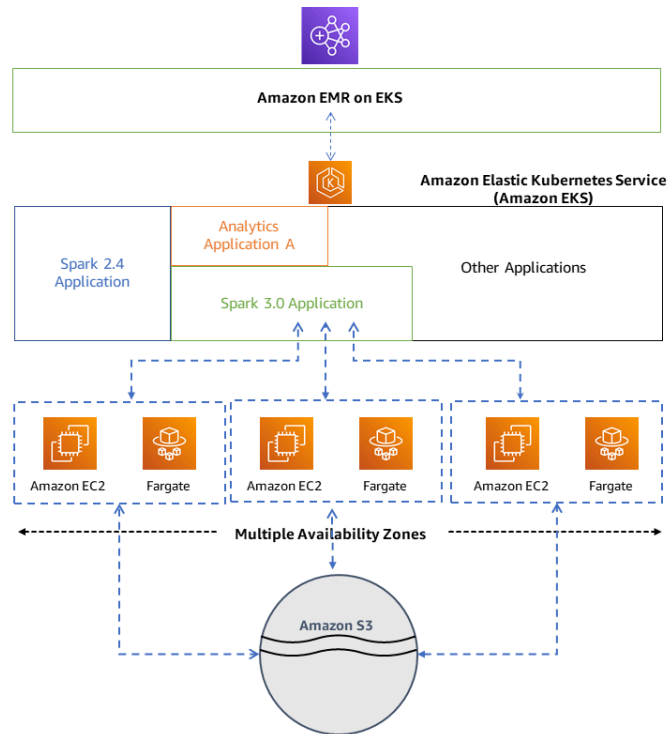
- [Arquitectura de Amazon EMR en EKS](#)
- [Descripción de conceptos y terminología de Amazon EMR en EKS](#)
- [Qué ocurre cuando envía trabajo a un clúster virtual de Amazon EMR en EKS](#)

Arquitectura de Amazon EMR en EKS

Amazon EMR en EKS acopla de forma flexible las aplicaciones a la infraestructura en la que se ejecutan. Cada capa de infraestructura proporciona orquestación para la capa siguiente. Cuando envía un trabajo a Amazon EMR, su definición de trabajo contiene todos los parámetros específicos de la aplicación. Amazon EMR utiliza estos parámetros para indicar a Amazon EKS qué pods y contenedores debe implementar. A continuación, Amazon EKS pone en línea los recursos informáticos EC2 de Amazon AWS Fargate necesarios para ejecutar el trabajo.

Con este acoplamiento flexible de servicios, puede ejecutar simultáneamente varios trabajos aislados de forma segura. También puede comparar el mismo trabajo con diferentes backends de computación o distribuir su trabajo en varias zonas de disponibilidad para mejorar la disponibilidad.

El siguiente diagrama ilustra cómo funciona Amazon EMR en EKS con otros AWS servicios.



Descripción de conceptos y terminología de Amazon EMR en EKS

Amazon EMR en EKS proporciona una opción de implementación para Amazon EMR que le permite ejecutar marcos de macrodatos de código abierto en Amazon Elastic Kubernetes Service (Amazon EKS). En este tema se proporciona un contexto sobre algunos de sus términos más comunes, como los espacios de nombres, los clústeres virtuales y las ejecuciones de trabajos, que son unidades de trabajo que se envían para su procesamiento.

Espacio de nombres de Kubernetes

Amazon EKS usa los espacios de nombres de Kubernetes para dividir los recursos del clúster entre varios usuarios y aplicaciones. Estos espacios de nombres son la base de los entornos de múltiples inquilinos. Un espacio de nombres de Kubernetes puede tener Amazon EC2 o AWS Fargate como proveedor de procesamiento. Esta flexibilidad le proporciona diferentes opciones de rendimiento y costo para ejecutar sus trabajos.

Clúster virtual

Un clúster virtual es un espacio de nombres de Kubernetes en el que Amazon EMR está registrado. Amazon EMR utiliza clústeres virtuales para ejecutar trabajos y alojar puntos de conexión. El mismo clúster físico puede respaldar varios clústeres virtuales. Sin embargo, cada clúster virtual se asigna a un espacio de nombres de un clúster de EKS. Los clústeres virtuales no crean ningún recurso activo que contribuya a su factura o que requiera una administración del ciclo de vida externa al servicio.

Ejecución de trabajo

Una ejecución de trabajo es una unidad de trabajo, como un jar de Spark, un PySpark script o una consulta de SparkSQL, que se envía a Amazon EMR en EKS. Un trabajo puede tener varias ejecuciones. Cuando envía una ejecución de trabajo, incluye la siguiente información:

- Un clúster virtual en el que debe ejecutarse el trabajo.
- Un nombre de trabajo para identificarlo.
- El rol de ejecución: un rol de IAM delimitado que ejecuta el trabajo y le permite especificar a qué recursos puede acceder el trabajo.
- La etiqueta de versión de Amazon EMR que especifica la versión de las aplicaciones de código abierto que se van a utilizar.
- Los artefactos que debe utilizar al enviar su trabajo, como los parámetros `spark-submit`.

De forma predeterminada, los registros se cargan en el servidor de Spark History y se puede acceder a ellos desde la AWS Management Console. También puede enviar registros de eventos, registros de ejecución y métricas a Amazon S3 y Amazon CloudWatch.

Contenedores de Amazon EMR

Contenedores de Amazon EMR es el [nombre de la API de Amazon EMR en EKS](#). El prefijo `emr-containers` se utiliza en las siguientes situaciones:

- Es el prefijo en los comandos de la CLI de Amazon EMR en EKS. Por ejemplo, `aws emr-containers start-job-run`.
- Es el prefijo antes de las acciones de la política de IAM de Amazon EMR en EKS. Por ejemplo, "Action": ["emr-containers:StartJobRun"]. Para obtener más información, consulte [Acciones de política de Amazon EMR en EKS](#).

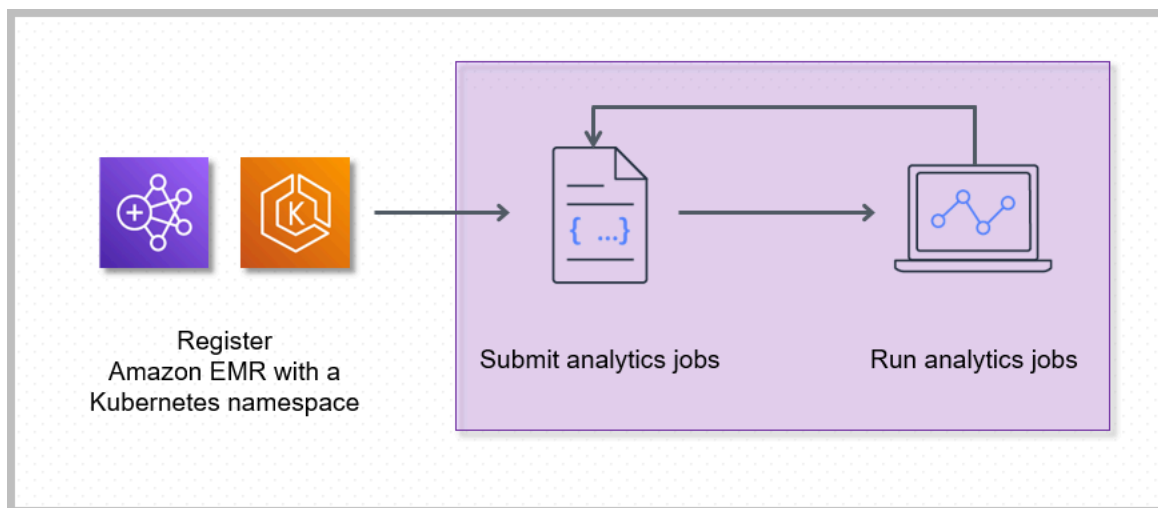
- Es el prefijo que se utiliza en los puntos de conexión de servicio de Amazon EMR en EKS. Por ejemplo, `emr-containers.us-east-1.amazonaws.com`. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Qué ocurre cuando envía trabajo a un clúster virtual de Amazon EMR en EKS

Al registrar Amazon EMR con un espacio de nombres de Kubernetes en Amazon EKS, se crea un clúster virtual. Amazon EMR puede entonces ejecutar cargas de trabajo de análisis en ese espacio de nombres. Cuando utiliza Amazon EMR en EKS para enviar trabajos de Spark al clúster virtual, Amazon EMR en EKS solicita al programador de Kubernetes de Amazon EKS que programe los pods.

Los siguientes pasos y diagrama ilustran el flujo de trabajo de Amazon EMR en EKS:

- Utilice un clúster de Amazon EKS existente o cree uno mediante la utilidad de línea de comandos [eksctl](#) o la consola de Amazon EKS.
- Para crear un clúster virtual, registre Amazon EMR con un espacio de nombres en un clúster EKS.
- Envíe su trabajo al clúster virtual mediante el SDK AWS CLI o el SDK.



Por cada trabajo que ejecuta, Amazon EMR en EKS crea un contenedor con una imagen base de Amazon Linux 2, Apache Spark y las dependencias asociadas. Cada trabajo se ejecuta en un pod que descarga el contenedor y comienza a ejecutarlo. El pod termina una vez terminado el trabajo. Si la imagen del contenedor se implementó previamente en el nodo, se utiliza una imagen almacenada

en caché y se omite la descarga. Los contenedores asociados, como los reenviadores de registros o métricas, se pueden implementar en el pod. Una vez finalizado el trabajo, podrá seguir depurándolo mediante la interfaz de usuario de la aplicación de Spark en la consola de Amazon EMR.

Introducción a Amazon EMR en EKS

Este tema le ayuda a empezar a utilizar Amazon EMR en EKS mediante la implementación de una aplicación de Spark en un clúster virtual. Incluye pasos para configurar los permisos correctos e iniciar un trabajo. Antes de comenzar, asegúrese de que ha realizado los pasos que se detallan en [Configuración de Amazon EMR en EKS](#). Esto le ayuda a obtener herramientas como la AWS CLI configuración antes de crear su clúster virtual. Para ver otras plantillas que pueden ayudarle a empezar, consulte nuestra [Guía de prácticas recomendadas para contenedores EMR](#) en GitHub

Necesitará la siguiente información de los pasos de configuración:

- ID de clúster virtual para el clúster de Amazon EKS y el espacio de nombres de Kubernetes registrado con Amazon EMR

Important

Al crear un clúster de EKS, asegúrese de utilizar m5.xlarge como tipo de instancia, o cualquier otro tipo de instancia con una CPU y una memoria superiores. El uso de un tipo de instancia con menos CPU o memoria que m5.xlarge puede provocar un error del trabajo debido a la insuficiencia de recursos disponibles en el clúster.

- Nombre del rol de IAM utilizado para la ejecución del trabajo
- Etiqueta de versión de la versión de Amazon EMR (por ejemplo, emr-6.4.0-latest)
- Objetivos de destino para el registro y la supervisión:
 - Nombre del grupo de CloudWatch registros de Amazon y prefijo de flujo de registro
 - Ubicación de Amazon S3 para almacenar registros de eventos y contenedores

Important

Los trabajos de Amazon EMR en EKS utilizan Amazon CloudWatch y Amazon S3 como destinos de destino para la supervisión y el registro. Puede supervisar el progreso del trabajo y solucionar los errores al consultar los registros de trabajos enviados a estos destinos. Para habilitar el registro, la política de IAM asociada con el rol de IAM para la ejecución del trabajo debe tener los permisos necesarios para acceder a los recursos de destino. Si la política de IAM no tiene los permisos necesarios, debe seguir los pasos descritos en [Actualizar la política de confianza del rol de ejecución de trabajos](#) [Configurar una ejecución de trabajo](#)

[para usar los registros de Amazon S3](#) y [Configurar una ejecución de trabajo para usar CloudWatch registros antes de ejecutar](#) este trabajo de muestra.

Ejecutar una aplicación de Spark

Siga estos pasos para ejecutar una aplicación de Spark en Amazon EMR en EKS. El archivo `entryPoint` de aplicación de una aplicación de Python de Spark se encuentra en `s3://REGION.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py`. **REGION** Es la región en la que reside su clúster virtual de Amazon EMR en EKS, por ejemplo. *us-east-1*

1. Actualice la política de IAM para el rol de ejecución de trabajos con los permisos necesarios, tal como se demuestra en las siguientes instrucciones de política.

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



```

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

```

- La primera instrucción `ReadFromLoggingAndInputScriptBuckets` de esta política concede acceso a `ListBucket` y `GetObject`s a los siguientes buckets de Amazon S3:
 - *REGION*.`elasticmapreduce`: el bucket en el que se encuentra el archivo `entryPoint` de la aplicación.
 - *amzn-s3-demo-destination-bucket*- un depósito que usted defina para sus datos de salida.
 - *amzn-s3-demo-logging-bucket*- un depósito que usted defina para sus datos de registro.

- La segunda instrucción de esta política otorga al trabajo permisos para escribir datos `WriteToLoggingAndOutputDataBuckets` en los buckets de salida y de registro, respectivamente.
 - La tercera declaración `DescribeAndCreateCloudwatchLogStream` otorga al trabajo permisos para describir y crear Amazon CloudWatch Logs.
 - La cuarta declaración `WriteToCloudwatchLogs` concede permisos para escribir registros en un grupo de CloudWatch registros de Amazon denominado *my_log_group_name* en un flujo de registros denominado *my_log_stream_prefix*.
2. Para ejecutar una aplicación de Python de Spark, use el siguiente comando. Sustituya todos los *red italicized* valores reemplazables por los valores adecuados. *REGION* Es la región en la que reside su clúster virtual de Amazon EMR en EKS, por ejemplo. *us-east-1*

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

Los datos de salida de este trabajo estarán disponibles en `s3://amzn-s3-demo-destination-bucket/wordcount_output`.

También puede crear un archivo JSON con parámetros especificados para la ejecución del trabajo. A continuación, ejecute el comando `start-job-run` con una ruta al archivo JSON. Para obtener más información, consulte [Enviar una ejecución de trabajo con StartJobRun](#). Para obtener más información sobre la configuración de los parámetros de ejecución del trabajo, consulte [Opciones para configurar una ejecución de trabajo](#).

3. Para ejecutar una aplicación de Spark SQL, use el siguiente comando. Sustituya todos los *red italicized* valores por los valores adecuados. *REGION* Es la región en la que reside su clúster virtual de Amazon EMR en EKS, por ejemplo. *us-east-1*

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

A continuación se muestra un archivo de consulta SQL de ejemplo. Debe tener un almacén de archivos externo, como S3, donde se almacenen los datos de las tablas.

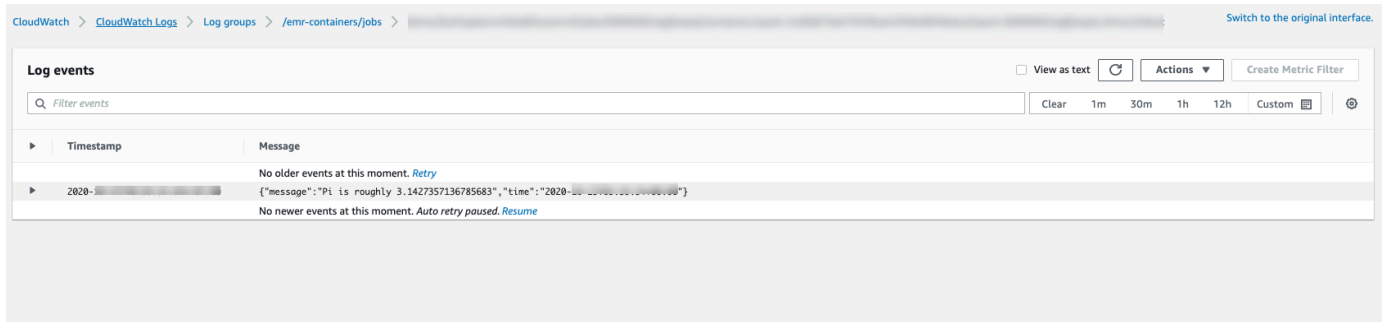
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
customer_id string, review_id string, product_id string, product_parent string,
product_title string, star_rating integer, helpful_votes integer, total_votes
integer, vine string, verified_purchase string,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

El resultado de este trabajo estará disponible en los registros estándar del controlador en S3 o CloudWatch, en función de la configuración `monitoringConfiguration` que esté configurada.

4. También puede crear un archivo JSON con parámetros especificados para la ejecución del trabajo. A continuación, ejecute el comando `start-job-run` con una ruta al archivo JSON. Para obtener más información, consulte [Enviar una ejecución de trabajo](#). Para obtener más información sobre la configuración de los parámetros de ejecución de trabajo, consulte [Opciones para configurar una ejecución de trabajo](#).

Para supervisar el progreso del trabajo o para depurar errores, puede inspeccionar los registros cargados en Amazon S3, en CloudWatch Logs o en ambos. Consulte la ruta de registro en Amazon S3 en [Configurar una ejecución de trabajo para usar registros de S3 y, para los registros](#) de Cloudwatch, en [Configurar una ejecución de trabajo para usar CloudWatch registros](#). Para ver los registros en los CloudWatch registros, siga las instrucciones que aparecen a continuación.

- Abre la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
- En el panel Navegación, elija Registros. A continuación, elija Grupos de registros.
- Elija el grupo de registros de Amazon EMR en EKS y, a continuación, consulte los eventos de registro cargados.



CloudWatch > CloudWatch Logs > Log groups > /emr-containers/jobs > [Switch to the original interface.](#)

Log events ☐ View as text [Actions](#) [Create Metric Filter](#)

[Clear](#) [1m](#) [30m](#) [1h](#) [12h](#) [Custom](#) [⌵](#) [⌵](#)

Timestamp	Message
	No older events at this moment. Retry
2020-03-10T14:27:35.7136785683Z	{"message": "PI is roughly 3.1427357136785683", "time": "2020-03-10T14:27:35.7136785683Z"}
	No newer events at this moment. Auto retry paused. Resume

Important

Los trabajos tienen una [política de reintentos configurada de forma predeterminada](#). Para obtener información sobre cómo modificar o deshabilitar la configuración, consulte [Uso de políticas de reintentos de trabajos](#).

Enlaces a las guías de prácticas recomendadas de Amazon EMR sobre EKS en GitHub

Hemos creado la [Guía de prácticas recomendadas de Amazon EMR en EKS](#) mediante la colaboración comunitaria de código abierto para poder iterar rápidamente y ofrecer recomendaciones para aspectos de creación y ejecución de un clúster virtual. Le recomendamos que utilice la [Guía de prácticas recomendadas de Amazon EMR en EKS](#) para las secciones. Elija los enlaces de cada sección para ir al GitHub sitio.

Seguridad

Note

Para obtener más información sobre la seguridad con Amazon EMR en EKS, consulte [Prácticas recomendadas de seguridad de Amazon EMR en EKS](#).

[Prácticas recomendadas de cifrado](#): cómo utilizar el cifrado para los datos en reposo y en tránsito.

En [Administración de la seguridad de la red](#) se describe cómo configurar grupos de seguridad para pods de Amazon EMR en EKS mientras se conecta a orígenes de datos alojados en Servicios de AWS , como Amazon RDS y Amazon Redshift.

[Uso del administrador de AWS secretos para almacenar secretos](#).

Envío de trabajos de PySpark

[Envío de trabajos de PySpark](#): especifica diferentes tipos de empaquetado para las aplicaciones de PySpark con formatos como zip, egg, wheel y pex.

Almacenamiento

[Uso de volúmenes de EBS](#): cómo utilizar el aprovisionamiento estático y dinámico para los trabajos que necesitan volúmenes de EBS.

[Uso de volúmenes de Amazon FSx for Lustre](#): cómo utilizar el aprovisionamiento estático y dinámico para los trabajos que necesitan volúmenes de Amazon FSx for Luster.

[Uso de volúmenes de almacenes de instancias](#): cómo utilizar los volúmenes de almacenes de instancias para el procesamiento de trabajos.

Integración con metaalmacenes

[Uso de metaalmacenes de Hive](#): ofrece diferentes formas de utilizar metaalmacenes de Hive.

[Uso de AWS Glue](#): ofrece diferentes formas de configurar el catálogo de AWS Glue.

Debugging

[Uso de la depuración de Spark](#): cómo cambiar el nivel de registro.

[Conexión a la interfaz de usuario de Spark en el pod controlador](#).

[Cómo utilizar el servidor de historial Spark autoalojado con Amazon EMR en EKS](#).

Solución de problemas de Amazon EMR en EKS

[Solución de problemas](#).

Colocación de nodos

[Uso de selectores de nodos de Kubernetes](#) para single-az y otros casos de uso.

[Uso de la colocación de nodos de Fargate](#).

Rendimiento

[Uso de la asignación dinámica de recursos \(DRA\)](#).

[Prácticas recomendadas de EKS](#) para el complemento Amazon VPC Container Network Interface (CNI), Cluster Autoscaler y Core DNS.

Optimización de costos

[Uso de instancias puntuales](#): prácticas recomendadas de instancias EC2 puntuales de Amazon y cómo utilizar la función de desmantelamiento de nodos de Spark.

¿Usando AWS Outposts

[Ejecución de Amazon EMR en EKS mediante AWS Outposts](#)

Personalización de las imágenes de Docker para Amazon EMR en EKS

Puede utilizar imágenes de Docker personalizadas con Amazon EMR en EKS. La personalización de la imagen de tiempo de ejecución de Amazon EMR en EKS ofrece las siguientes ventajas:

- Empaquete las dependencias de las aplicaciones y el entorno de tiempo de ejecución en un único contenedor inmutable que promueva la portabilidad y simplifique la administración de dependencias para cada carga de trabajo.
- Instale y configure paquetes optimizados para sus cargas de trabajo. Es posible que estos paquetes no estén ampliamente disponibles en la distribución pública de los tiempos de ejecución de Amazon EMR.
- Integre Amazon EMR en EKS con los procesos de compilación, prueba e implementación establecidos actualmente en su organización, incluido el desarrollo y las pruebas locales.
- Aplique procesos de seguridad establecidos, como el escaneo de imágenes, que cumplan con los requisitos de cumplimiento y gobernanza de su organización.

Temas

- [Cómo personalizar las imágenes de Docker](#)
- [Información sobre cómo seleccionar un URI de imagen base](#)
- [Consideraciones para personalizar imágenes](#)

Cómo personalizar las imágenes de Docker

Siga estos pasos para personalizar las imágenes de Docker de Amazon EMR en EKS. Los pasos muestran cómo obtener una imagen base, personalizarla y publicarla, y cómo enviar una carga de trabajo mediante la imagen.

- [Requisitos previos](#)
- [Paso 1: recuperar una imagen base de Amazon Elastic Container Registry \(Amazon ECR\)](#)
- [Paso 2: personalizar una imagen base](#)
- [Paso 3: \(opcional, pero recomendado\) validar una imagen personalizada](#)
- [Paso 4: publicar una imagen personalizada](#)

- [Paso 5: enviar una carga de trabajo de Spark en Amazon EMR mediante una imagen personalizada](#)

Note

Otras opciones que quizás desee tener en cuenta al personalizar las imágenes de Docker son la personalización para puntos de conexión interactivos, lo que se hace para garantizar que cuenta con las dependencias necesarias, o el uso de imágenes de contenedor con varias arquitecturas:

- [Personalice las imágenes de Docker para puntos de conexión interactivos](#)
- [Uso de imágenes multiarquitectura](#)

Requisitos previos

- Complete los pasos [Configuración de Amazon EMR en EKS](#) de Amazon EMR en EKS.
- Instale Docker en su entorno. Para obtener más información, consulte [Obtener Docker](#).

Paso 1: recuperar una imagen base de Amazon Elastic Container Registry (Amazon ECR)

La imagen base contiene el tiempo de ejecución de Amazon EMR y los conectores que se utilizan para acceder a otros servicios de AWS . Para Amazon EMR 6.9.0 y versiones posteriores, puede obtener las imágenes base en Amazon ECR Public Gallery. Navegue por la galería para encontrar el enlace a la imagen y llevarla a su espacio de trabajo local. Por ejemplo, para la versión 7.3.0 de Amazon EMR, el siguiente `docker pull` comando `docker pull` le brinda la imagen base estándar más reciente. Puede sustituir `emr-7.8.0:latest` por `emr-7.8.0-spark-rapids:latest` para recuperar la imagen que tiene el acelerador de Nvidia RAPIDS. También puede sustituir `emr-7.8.0:latest` por `emr-7.8.0-java11:latest` para recuperar la imagen con el tiempo de ejecución de Java 11.

```
docker pull public.ecr.aws/emr-on-eks/spark/emr-7.8.0:latest
```

Si desea recuperar la imagen base de Amazon EMR 6.9.0 o versiones anteriores, o si prefiere recuperarla de las cuentas de registro de Amazon ECR de cada región, siga estos pasos:

1. Elija un URI de imagen base. El URI de imagen sigue este formato, *ECR-registry-account.dkr.ecr.Region.amazonaws.com/spark/container-image-tag*, tal como se muestra en el siguiente ejemplo.

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

Para elegir una imagen base en su región, consulte [Información sobre cómo seleccionar un URI de imagen base](#).

2. Inicie sesión en el repositorio de Amazon ECR donde está almacenada la imagen base. Sustituya *895885662937 us-west-2* emr6.00 como la imagen de Amazon EMR y la AWS región de Amazon EMR.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 895885662937.dkr.ecr.us-west-2.amazonaws.com
```

3. Coloque la imagen base en su espacio de trabajo local. Sustituya *emr-6.6.0:latest* emr6.0:6.0:6.0:6.0:6.0:6.0:6.0:

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

Paso 2: personalizar una imagen base

Siga estos pasos para personalizar la imagen base que ha obtenido de Amazon ECR.

1. Cree un Dockerfile nuevo en su espacio de trabajo local.
2. Edite el Dockerfile que acaba de crear y agregue el siguiente contenido. Este Dockerfile usa la imagen del contenedor de la que ha extraído de *895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest*.

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

3. Agregue comandos en el Dockerfile para personalizar la imagen base. Por ejemplo, agregue un comando para instalar bibliotecas de Python, como se muestra en el siguiente Dockerfile.

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

```
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

- Desde el mismo directorio en el que Dockerfile se creó, ejecute el siguiente comando para crear la imagen de Docker. Proporcione un nombre para la imagen de Docker, por ejemplo, `emr6.6_custom`. *emr6.6_custom*

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

Paso 3: (opcional, pero recomendado) validar una imagen personalizada

Le recomendamos que pruebe la compatibilidad de la imagen personalizada antes de publicarla. Puede utilizar la [CLI de imágenes personalizadas de Amazon EMR en EKS](#) para comprobar si la imagen tiene las estructuras de archivos requeridas y las configuraciones correctas para ejecutarse en Amazon EMR en EKS.

Note

La CLI de imagen personalizada de Amazon EMR en EKS no puede confirmar que la imagen no contenga errores. Tenga cuidado al eliminar las dependencias de las imágenes base.

Siga estos pasos para validar la imagen personalizada.

- Descargue e instale Amazon EMR en EKS en la CLI de imágenes personalizadas. Para obtener más información, consulte la [Guía de instalación de la CLI de imagen personalizada de Amazon EMR en EKS](#).
- Ejecute el siguiente comando para evaluar la instalación.

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

A continuación se muestra un ejemplo de este resultado.

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

- Ejecute el siguiente comando para validar la imagen personalizada.

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

- `-i` especifica el URI de imagen local que debe validarse. Puede ser el URI de imagen o cualquier nombre o etiqueta que haya definido para la imagen.
- `-r` especifica la versión de lanzamiento exacta de la imagen base, por ejemplo, `emr-6.6.0-latest`.
- `-t` especifica el tipo de imagen. Si se trata de una imagen de Spark, ingrese `spark`. El valor predeterminado es `spark`. La versión actual de la CLI de imágenes personalizadas de Amazon EMR en EKS solo admite imágenes en tiempo de ejecución de Spark.

Si ejecuta el comando correctamente y la imagen personalizada cumple con todas las configuraciones y estructuras de archivos requeridas, el resultado devuelto muestra los resultados de todas las pruebas, tal y como se muestra en el siguiente ejemplo.

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

Si la imagen personalizada no cumple con las configuraciones o estructuras de archivos requeridas, aparecen mensajes de error. El resultado devuelto proporciona información sobre las configuraciones o estructuras de archivos incorrectas.

Paso 4: publicar una imagen personalizada

Publique la nueva imagen de Docker en su registro de Amazon ECR.

1. Ejecute el siguiente comando para crear un repositorio de Amazon ECR para almacenar la imagen de Docker. Proporcione un nombre para la imagen de Docker, por ejemplo, `emr6.6_custom`*emr6.6_custom_repo*. Reemplace la *us-west-2* por su región.

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

Para obtener más información, consulte [Crear un repositorio](#) en la Guía del usuario de Amazon ECR.

2. Ejecute el siguiente comando para autenticarse en el registro predeterminado.

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

Para obtener más información, consulte [Autenticar en su registro predeterminado](#) en la Guía del usuario de Amazon ECR.

3. Etiquete y publique una imagen en el repositorio de Amazon ECR que ha creado.

Etiquete la imagen.

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

Inserte la imagen.

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

Para obtener más información, consulte [Insertar una imagen en Amazon ECR](#) en la Guía del usuario de Amazon ECR.

Paso 5: enviar una carga de trabajo de Spark en Amazon EMR mediante una imagen personalizada

Una vez creada y publicada una imagen personalizada, puede enviar un trabajo de Amazon EMR en EKS mediante una imagen personalizada.

En primer lugar, cree un archivo start-job-run-request JSON de ejemplo.

`spark.kubernetes.container.image`

Note

Puede usar el esquema `local://` para hacer referencia a los archivos disponibles en la imagen personalizada, tal como se muestra con el argumento `entryPoint` en el siguiente fragmento de código JSON. También puede usar el esquema `local://` para hacer referencia a las dependencias de las aplicaciones. Todos los archivos y dependencias a los que se hace referencia mediante el esquema `local://` ya deben estar presentes en la ruta especificada en la imagen personalizada.

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

}

También puede hacer referencia a la imagen personalizada con las propiedades `applicationConfiguration`, tal como se muestra en el siguiente ejemplo.

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

A continuación, ejecute el comando `start-job-run` para enviar el trabajo.

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

[illegible]

más recientes. Para obtener más información sobre las versiones de lanzamiento de Amazon EMR y sus etiquetas de imagen, consulte [Información sobre cómo seleccionar un URI de imagen base](#).

 Note

Puede usar `spark.kubernetes.driver.container.image` y `spark.kubernetes.executor.container.image` para especificar una imagen diferente para los pods controladores y ejecutores.

Personalice las imágenes de Docker para puntos de conexión interactivos

También puede personalizar las imágenes de Docker de puntos de conexión interactivos, de modo que pueda ejecutar imágenes base del kernel personalizadas. Esto le ayuda a garantizar que disponga de las dependencias que necesita al ejecutar cargas de trabajo interactivas desde EMR Studio.

1. Siga los [pasos 1-4](#) descritos anteriormente para personalizar una imagen de Docker. Para las versiones 6.9.0 y posteriores de Amazon EMR, puede obtener el URI de imagen base en Amazon ECR Public Gallery. Para las versiones anteriores a Amazon EMR 6.9.0, puede obtener la imagen en las cuentas de Amazon ECR Registry de cada Región de AWS, y la única diferencia es el URI de la imagen base de su archivo de Docker. El URI de imagen base sigue el siguiente formato:

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

Debe usar `notebook-spark` en el URI de imagen base en lugar de `spark`. La imagen base contiene el tiempo de ejecución de Spark y los kernels del cuaderno que se ejecutan con él. Para obtener más información sobre cómo seleccionar las regiones y las etiquetas de imagen de contenedores, consulte [Información sobre cómo seleccionar un URI de imagen base](#).

 Note

Actualmente, solo se admiten anulaciones de imágenes base y no se admite la introducción de kernels completamente nuevos de otros tipos distintos de las imágenes base que AWS proporciona.

2. Cree un punto de conexión interactivo que se pueda utilizar con la imagen personalizada.

Primero, cree un archivo JSON denominado `custom-image-managed-endpoint.json` con el siguiente contenido.

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

A continuación, cree un punto de conexión interactivo con las configuraciones especificadas en el archivo JSON, tal como se muestra en el siguiente ejemplo.

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

Para obtener más información, consulte [Crear un punto de conexión interactivo para su clúster virtual](#).

3. Conéctese al punto de conexión interactivo a través de EMR Studio. Para obtener más información, consulte [Conexión desde Studio](#).

Uso de imágenes multiarquitectura

Amazon EMR en EKS es compatible con imágenes de contenedor multiarquitectura de Amazon Elastic Container Registry (Amazon ECR). Para obtener más información, consulte [Introducción a las imágenes de contenedores de varias arquitecturas de Amazon ECR](#).

Las imágenes personalizadas de Amazon EMR en EKS admiten tanto instancias de EC2 basadas en Graviton como instancias de EC2 AWS no basadas en Graviton EC2 . non-Graviton-based EC2 Las imágenes basadas en Graviton se almacenan en los mismos repositorios de imágenes de Amazon ECR que las imágenes no basadas en Graviton. non-Graviton-based

Por ejemplo, para inspeccionar la lista de manifiesto de Docker en busca de imágenes de 6.6.0, ejecute el siguiente comando.

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

Esta es la salida. La arquitectura arm64 es para la instancia de Graviton. amd64 es para una instancia que no es de Graviton.

```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
  "manifests": [
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 1805,
      "digest":
"xxx123:6b971cb47d11011ab3d45fff925e9442914b4977ae0f9fbcdf5cfa99a7593f0",
      "platform": {
        "architecture": "arm64",
        "os": "linux"
      }
    },
  ],
}
```

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

Siga estos pasos para crear imágenes multiarquitectura:

1. Cree un Dockerfile con el siguiente contenido para poder extraer la imagen arm64.

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

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

2. Para crear una imagen multiarquitectura en Amazon ECR, siga las instrucciones de [Introducción a las imágenes de contenedores de varias arquitecturas de Amazon ECR](#).

 Note

Debe crear imágenes arm64 en las instancias arm64. Del mismo modo, debe crear imágenes amd64 en las instancias amd64.

También puede crear imágenes de varias arquitecturas sin tener que basarse en cada tipo de instancia específico con el comando `buildx` de Docker. Para obtener más información, consulte [Aprovechar la compatibilidad con arquitectura de múltiples CPU](#).

3. Tras crear la imagen multiarquitectura, puede enviar un trabajo con el mismo parámetro `spark.kubernetes.container.image` y dirigirlo a la imagen. En un clúster heterogéneo con non-Graviton-based EC2 instancias de EC2 AWS basadas en Graviton y no basadas en Graviton, la instancia determina la imagen de arquitectura de la instancia que extrae la imagen.

Información sobre cómo seleccionar un URI de imagen base

Note

Para Amazon EMR 6.9.0 y versiones posteriores, puede recuperar la imagen base de Amazon ECR Public Gallery, por lo que no necesita crear el URI de imagen base como se indica en las instrucciones de esta página. Para encontrar la etiqueta de imagen de contenedor para su imagen base, consulte la [página de notas de la versión](#) de la versión correspondiente de Amazon EMR en EKS.

Las imágenes de Docker base que puede seleccionar se almacenan en Amazon Elastic Container Registry (Amazon ECR). El URI de imagen sigue este formato: *ECR-registry-account.dkr.ecr.Region.amazonaws.com/spark/container-image-tag*, tal como se muestra en el siguiente ejemplo.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.8.0:latest
```

El URI de imagen de los puntos de conexión interactivos sigue este formato: *ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag*, tal como se muestra en el siguiente ejemplo. Debe usar notebook-spark en el URI de imagen base en lugar de spark.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-7.8.0:latest
```

Del mismo modo, para las imágenes de python3 que no son de Spark para puntos de conexión interactivos, el URI de imagen es *ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-python/container-image-tag*. El siguiente ejemplo de URI tiene el formato correcto:

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-7.8.0:latest
```

Para encontrar la etiqueta de imagen de contenedor para su imagen base, consulte la [página de notas de la versión](#) de la versión correspondiente de Amazon EMR en EKS.

Cuentas de registro de Amazon ECR por región

Para evitar una alta latencia de red, extraiga una imagen base de su más cercana Región de AWS. Seleccione la cuenta de registro de Amazon ECR que corresponda con la región de la que extrae la imagen según la siguiente tabla.

Regiones	Cuentas de registro de Amazon ECR	
ap-east-1	736135916053	
ap-northeast-1	059004520145	
ap-northeast-2	996579266876	
ap-northeast-3	705689932349	
ap-southeast-3	946962994502	
ap-south-1	235914868574	
ap-south-2	691480105545	
ap-southeast-1	671219180197	
ap-southeast-2	038297999601	
ca-central-1	351826393999	
eu-central-1	107292555468	
eu-central-2	314408114945	
eu-north-1	830386416364	
eu-west-1	483788554619	
eu-west-2	118780647275	
eu-west-3	307523725174	

Regiones	Cuentas de registro de Amazon ECR	
eu-south-1	238014973495	
eu-south-2	350796622945	
il-central-1	395734710648	
me-south-1	008085056818	
me-central-1	818935616732	
sa-east-1	052806832358	
us-gov-west-1	299385240661	
us-gov-east-1	299393998622	
us-east-1	755674844232	
us-east-2	711395599931	
us-west-1	608033475327	
us-west-2	895885662937	
af-south-1	358491847878	
cn-north-1	068337069695	
cn-northwest-1	068420816659	

Consideraciones para personalizar imágenes

Al personalizar las imágenes de Docker, puede elegir el tiempo de ejecución exacto para su trabajo a un nivel granular. Tenga en cuenta estas prácticas recomendadas al utilizar esta característica. Estas incluyen consideraciones de seguridad, configuración y montaje de una imagen:

- La seguridad es una responsabilidad compartida entre AWS y usted. Es responsable de aplicar los parches de seguridad a los archivos binarios que agregue a la imagen. Siga las [Prácticas recomendadas de seguridad de Amazon EMR en EKS](#), especialmente [Obtener las actualizaciones de seguridad más recientes para imágenes personalizadas](#) y [Aplicar el principio de privilegio mínimo](#).
- Al personalizar una imagen base, debe cambiar el usuario de Docker a `hadoop:hadoop` para que los trabajos no se ejecuten con el usuario raíz.
- Amazon EMR en EKS monta los archivos sobre las configuraciones de la imagen, como `spark-defaults.conf`, en tiempo de ejecución. Para anular estos archivos de configuración, le recomendamos que utilice el parámetro `applicationOverrides` durante el envío del trabajo y no modifique directamente los archivos de la imagen personalizada.
- Amazon EMR en EKS monta determinadas carpetas en tiempo de ejecución. Las modificaciones que haga en estas carpetas no están disponibles en el contenedor. Si desea agregar una aplicación o sus dependencias para imágenes personalizadas, le recomendamos que elija un directorio que no forme parte de las siguientes rutas predefinidas:
 - `/var/log/fluentd`
 - `/var/log/spark/user`
 - `/var/log/spark/apps`
 - `/mnt`
 - `/tmp`
 - `/home/hadoop`
- Puede cargar su imagen personalizada en cualquier repositorio compatible con Docker, como Amazon ECR, Docker Hub o un repositorio empresarial privado. Para obtener más información sobre cómo configurar la autenticación del clúster de Amazon EKS con el repositorio de Docker seleccionado, consulte [Extraer una imagen de un registro privado](#).

Ejecución de trabajos de Flink con Amazon EMR en EKS

Las versiones 6.13.0 y posteriores de Amazon EMR admiten Amazon EMR en EKS con Apache Flink, o el operador de Flink Kubernetes, como modelo de envío de trabajos para Amazon EMR en EKS. Con Amazon EMR en EKS con Apache Flink, puede implementar y administrar aplicaciones de Flink con el tiempo de ejecución de versiones de Amazon EMR en sus propios clústeres de Amazon EKS. Una vez que haya implementado el operador de Flink Kubernetes en su clúster de Amazon EKS, podrá enviar las solicitudes de Flink directamente al operador. El operador administra el ciclo de vida de las aplicaciones de Flink.

Temas

- [Configuración y uso del operador de Flink Kubernetes](#)
- [Uso de Kubernetes nativo de Flink](#)
- [Personalización de las imágenes de Docker para Flink y FluentD](#)
- [Supervisión del operador de Flink Kubernetes y de los trabajos de Flink](#)
- [Cómo apoya Flink la alta disponibilidad y la resiliencia al trabajo](#)
- [Uso del escalador automático para aplicaciones de Flink](#)
- [Mantenimiento y solución de problemas para trabajos de Flink en Amazon EMR en EKS](#)
- [Versiones de Amazon EMR en EKS compatibles con Apache Flink](#)

Configuración y uso del operador de Flink Kubernetes

En las siguientes páginas, se describe cómo configurar y usar el operador de Kubernetes de Flink para ejecutar trabajos de Flink con Amazon EMR en EKS. Los temas disponibles incluyen los requisitos previos necesarios, cómo configurar el entorno y cómo ejecutar una aplicación Flink en Amazon EMR en EKS.

Temas

- [Configuración del operador de Kubernetes de Flink para Amazon EMR en EKS](#)
- [Instalación del operador de Flink Kubernetes de Amazon EMR en EKS](#)
- [Ejecutar una aplicación de Flink](#)
- [Permisos de rol de seguridad para ejecutar una aplicación Flink](#)
- [Desinstalación del operador de Kubernetes de Flink de Amazon EMR en EKS](#)

Configuración del operador de Kubernetes de Flink para Amazon EMR en EKS

Complete las siguientes tareas para configurarlo todo antes de instalar el operador de Flink Kubernetes en Amazon EKS. Si ya se registró en Amazon Web Services (AWS) y ha usado Amazon EKS, lo tiene todo casi listo para comenzar a utilizar Amazon EMR en EKS. Complete las siguientes tareas para la configuración del operador de Flink en Amazon ECS. Si ya ha completado alguno de los requisitos previos, puede omitirlos y pasar al siguiente.

- [Instalar o actualizar a la última versión de AWS CLI](#): si ya ha instalado el AWS CLI, confirme que tiene la última versión.
- [Configure kubectl y eksctl](#): eksctl es una herramienta de línea de comandos que se utiliza para comunicarse con Amazon EKS.
- [Instale Helm](#): el administrador de paquetes Helm para Kubernetes le ayuda a instalar y administrar aplicaciones en el clúster de Kubernetes.
- [Comience con Amazon EKS: eksctl](#) : siga los pasos para crear un nuevo clúster de Kubernetes con nodos en Amazon EKS.
- [Elija una etiqueta de versión de Amazon EMR](#) (versión 6.13.0 o posterior): el operador de Flink Kubernetes es compatible con las versiones 6.13.0 y posteriores de Amazon EMR.
- [Habilite los roles de IAM para las cuentas de servicio \(IRSA\) en el clúster de Amazon EKS](#).
- [Cree un rol de ejecución de trabajos](#).
- [Actualice la política de confianza del rol de ejecución de trabajos](#).
- Cree un rol de ejecución de operador. Este paso es opcional. Puede usar el mismo rol para los trabajos y el operador de Flink. Si desea que su operador tenga un rol de IAM diferente, puede crear un rol independiente.
- Actualice la política de confianza del rol de ejecución del operador. Debe agregar explícitamente una entrada de política de confianza para los roles que desee utilizar para la cuenta de servicio del operador de Flink Kubernetes de Amazon EMR. Puede seguir este formato de ejemplo:

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

```

        "Federated": "arn:aws:iam::ACCOUNT_ID:oidc-provider/OIDC_PROVIDER"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
        "StringLike": {
            "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-
containers-sa-flink-operator"
        }
    }
}
]
}

```

Instalación del operador de Flink Kubernetes de Amazon EMR en EKS

Este tema le ayuda a empezar a utilizar el operador de Flink Kubernetes en Amazon EKS mediante una implementación de Flink.

Instalación del operador de Kubernetes

Siga estos pasos para instalar el operador de Kubernetes para Apache Flink.

1. Si aún no lo ha hecho, complete los pasos de [the section called “Configuración”](#).
2. Instale *cert-manager* (una vez por clúster de Amazon EKS) para habilitar la incorporación del componente webhook.

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

3. Instale el gráfico de Helm.

```

export VERSION=7.8.0 # The Amazon EMR release version
export NAMESPACE=The Kubernetes namespace to deploy the operator

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

```

Ejemplo de salida:

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

4. Espere a que se complete la implementación y verifique la instalación del gráfico.

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

5. Debería ver el siguiente mensaje cuando se complete la implementación.

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

6. Use el siguiente comando para ver el operador implementado.

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

A continuación, se muestra un ejemplo de resultado, en el que la versión de la aplicación `x.y.z-amzn-n` se correspondería con la versión del operador de Flink para su versión de Amazon EMR en EKS. Para obtener más información, consulte [Versiones de Amazon EMR en EKS compatibles con Apache Flink](#).

NAME		NAMESPACE	REVISION	UPDATED	
	STATUS	CHART			APP VERSION
flink-kubernetes-operator		\$NAMESPACE	1	2023-02-22 16:43:45.24148	
-0500 EST	deployed	flink-kubernetes-operator-emr-7.8.0			x.y.z-amzn-n

Ejecutar una aplicación de Flink

A partir de Amazon EMR 6.13.0, puede ejecutar una aplicación Flink con el operador Flink Kubernetes en el modo de aplicación de Amazon EMR en EKS. Además, a partir de Amazon EMR 6.15.0, puede ejecutar una aplicación de Flink en el modo de sesiones. En esta página, se explican los dos métodos con los que puede ejecutar una aplicación de Flink con Amazon EMR en EKS.

Note

Debe tener un bucket de Amazon S3 creado para almacenar los metadatos de alta disponibilidad cuando envíe su trabajo de Flink. Si no desea usar esta característica, puede desactivarla. Está habilitada de forma predeterminada.

Requisito previo: antes de poder ejecutar una aplicación de Flink con el operador de Flink Kubernetes, complete los pasos indicados en [the section called “Configuración”](#) y [the section called “Instalación del operador de Kubernetes”](#).

Application mode

A partir de Amazon EMR 6.13.0, puede ejecutar una aplicación Flink con el operador Flink Kubernetes en el modo de aplicación de Amazon EMR en EKS.

1. Cree un archivo `basic-example-app-cluster.yaml` de definición de FlinkDeployment como en el siguiente ejemplo. Si activó y usó una de las [opciones de Regiones de AWS](#), asegúrese de quitar los comentarios y establecer la configuración `fs.s3a.endpoint.region`.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-app-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.13.0-flink-latest" # 6.13 or higher
  jobManager:
    storageDir: HIGH_AVAILABILITY_STORAGE_PATH
  resource:
    memory: "2048m"
    cpu: 1
  taskManager:
    resource:
```

```

    memory: "2048m"
    cpu: 1
  job:
    # if you have your job jar in S3 bucket you can use that path as well
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
    parallelism: 2
    upgradeMode: savepoint
    savepointTriggerNonce: 0
  monitoringConfiguration:
    cloudWatchMonitoringConfiguration:
      logGroupName: LOG_GROUP_NAME

```

- Envíe la implementación de Flink con el siguiente comando. Esto también creará un objeto de FlinkDeployment llamado basic-example-app-cluster.

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

- Acceda a la interfaz de usuario de Flink.

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

- Abra localhost:8081 para ver sus trabajos de Flink de forma local.
- Limpie el trabajo. Recuerde limpiar los artefactos de S3 que se crearon para este trabajo, como los puntos de control, la alta disponibilidad, los metadatos de puntos de guardado y los registros de CloudWatch. CloudWatch

Para obtener más información sobre cómo enviar solicitudes a Flink a través del operador de Kubernetes de Flink, consulte los ejemplos del operador de Kubernetes de [Flink en la carpeta apaches](#). apache/flink-kubernetes-operator GitHub

Session mode

A partir de Amazon EMR 6.15.0, puede ejecutar una aplicación Flink con el operador Flink Kubernetes en el modo de sesiones de Amazon EMR en EKS.

- Cree un archivo de definición FlinkDeployment denominado basic-example-app-cluster.yaml como en el siguiente ejemplo. Si activó y usó una de las [opciones de Regiones de AWS, asegúrese de quitar](#) los comentarios y establecer la configuración fs.s3a.endpoint.region. fs.s3a.endpoint.region

```

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

```

- Envíe la implementación de Flink con el siguiente comando. Esto también creará un objeto de FlinkDeployment llamado basic-example-session-cluster.

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

- Utilice el siguiente comando para confirmar que el clúster de sesión LIFECYCLE sea STABLE:

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

La salida debería tener un aspecto similar al siguiente ejemplo:

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

4. Cree un archivo de recursos de definición personalizado para un `FlinkSessionJob` con el nombre `basic-session-job.yaml` y el siguiente contenido:

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

5. Envíe la sesión de trabajo de Flink con el siguiente comando. Esto también creará el objeto de `FlinkSessionJob` con el nombre `basic-session-job`.

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

6. Utilice el siguiente comando para confirmar que el clúster de sesión LIFECYCLE sea STABLE y que JOB STATUS sea RUNNING:

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

La salida debería tener un aspecto similar al siguiente ejemplo:

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

7. Acceda a la interfaz de usuario de Flink.


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

8. Abra `localhost:8081` para ver sus trabajos de Flink de forma local.
9. Limpie el trabajo. Recuerde limpiar los artefactos de S3 que se crearon para este trabajo, como los puntos de control, la alta disponibilidad, los metadatos de puntos de guardado y los registros de CloudWatch. CloudWatch

Permisos de rol de seguridad para ejecutar una aplicación Flink

En este tema se describen los roles de seguridad para implementar y ejecutar una aplicación Flink. Hay dos roles necesarios para administrar una implementación y para crear y administrar trabajos: el rol de operador y el rol de trabajo. En este tema se presentan y se enumeran sus permisos.

Control de acceso basado en roles

Para implementar el operador y ejecutar los trabajos de Flink, se deben crear dos roles en Kubernetes: un rol de operador y otro de trabajo. Amazon EMR crea los dos roles de forma predeterminada al instalar el operador.

Rol de operador

Usamos el rol de operador `flinkdeployments JobManager` para cada trabajo de Flink y otros recursos de Flink y otros recursos, como servicios.

El nombre predeterminado del rol de operador es `emr-containers-sa-flink-operator` y requiere los siguientes permisos.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - services
  - events
  - configmaps
  - secrets
  - serviceaccounts
  verbs:
  - '*'
- apiGroups:
```

```
- rbac.authorization.k8s.io
resources:
- roles
- rolebindings
verbs:
- '*'
- apiGroups:
- apps
resources:
- deployments
- deployments/finalizers
- replicasets
verbs:
- '*'
- apiGroups:
- extensions
resources:
- deployments
- ingresses
verbs:
- '*'
- apiGroups:
- flink.apache.org
resources:
- flinkdeployments
- flinkdeployments/status
- flinksessionjobs
- flinksessionjobs/status
verbs:
- '*'
- apiGroups:
- networking.k8s.io
resources:
- ingresses
verbs:
- '*'
- apiGroups:
- coordination.k8s.io
resources:
- leases
verbs:
- '*'
```

Rol de trabajo

JobManager Utiliza el rol de trabajo para crear y administrar TaskManagers y ConfigMaps para cada trabajo.

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

Desinstalación del operador de Kubernetes de Flink de Amazon EMR en EKS

Siga estos pasos para desinstalar el operador de Kubernetes de Flink.

1. Elimine el operador.

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

2. Elimine los recursos de Kubernetes que Helm no desinstale.

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

3. (Opcional) Elimine cert-manager.

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

Uso de Kubernetes nativo de Flink

Las versiones 6.13.0 y posteriores de Amazon EMR admiten Kubernetes nativo de Flink como herramienta de línea de comandos que puede utilizar para enviar y ejecutar aplicaciones de Flink en un clúster de Amazon EMR en EKS.

Temas

- [Configuración de Kubernetes nativo de Flink para Amazon EMR en EKS](#)
- [Introducción a Kubernetes nativo de Flink para Amazon EMR en EKS](#)
- [Requisitos de seguridad de la cuenta de JobManager servicio de Flink para Kubernetes nativos](#)

Configuración de Kubernetes nativo de Flink para Amazon EMR en EKS

Complete las siguientes tareas para llevar a cabo la configuración antes de poder ejecutar una aplicación con la CLI de Flink en Amazon EMR en EKS. Si ya se registró en Amazon Web Services (AWS) y ha usado Amazon EKS, lo tiene todo casi listo para comenzar a utilizar Amazon EMR en EKS. Si ya ha completado alguno de los requisitos previos, puede omitirlos y pasar al siguiente.

- [Instalar o actualizar a la última versión de AWS CLI](#): si ya ha instalado el AWS CLI, confirme que tiene la última versión.
- [Comience con Amazon EKS: eksctl](#) : siga los pasos para crear un nuevo clúster de Kubernetes con nodos en Amazon EKS.
- [Seleccione un URI de imagen base de Amazon EMR](#) (versión 6.13.0 o posterior): el comando de Kubernetes de Flink es compatible con las versiones 6.13.0 y posteriores de Amazon EMR.
- Confirma que la cuenta JobManager de servicio tiene los permisos adecuados para crear y ver los TaskManager pods. Para obtener más información, consulta los [requisitos de seguridad de la cuenta JobManager de servicio de Flink para Kubernetes nativo](#).
- Configure su [perfil de credenciales de AWS](#) local.
- [Cree o actualice un archivo kubeconfig para un clúster de Amazon EKS](#) en el que desee ejecutar las aplicaciones de Flink.

Introducción a Kubernetes nativo de Flink para Amazon EMR en EKS

Estos pasos le muestran cómo configurar, ajustar una cuenta de servicio y ejecutar una aplicación de Flink. El Kubernetes nativo de Flink se usa para implementar Flink en un clúster de Kubernetes en ejecución.

Configuración y ejecución de una aplicación de Flink

Amazon EMR 6.13.0 y versiones posteriores es compatible con Kubernetes nativo de Flink para ejecutar aplicaciones de Flink en un clúster de Amazon EKS. Para ejecutar una aplicación de Flink, siga estos pasos:

1. Para poder ejecutar una aplicación de Flink con el comando Kubernetes nativo de Flink, complete los pasos que se indican en [the section called “Configuración”](#).
2. [Descargue e instale Flink](#).
3. Establezca los valores para las siguientes variables de entorno.

```
#Export the FLINK_HOME environment variable to your local installation of Flink
export FLINK_HOME=/usr/local/bin/flink #Will vary depending on your installation
export NAMESPACE=flink
export CLUSTER_ID=flink-application-cluster
export IMAGE=<123456789012.dkr.ecr.sample-Región de AWS-.amazonaws.com/flink/
emr-6.13.0-flink:latest>
export FLINK_SERVICE_ACCOUNT=emr-containers-sa-flink
export FLINK_CLUSTER_ROLE_BINDING=emr-containers-crb-flink
```

4. Cree una cuenta de servicio para administrar los recursos de Kubernetes.

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

5. Ejecute el comando run-application de la CLI.

```
$FLINK_HOME/bin/flink run-application \
  --target kubernetes-application \
  -Dkubernetes.namespace=$NAMESPACE \
  -Dkubernetes.cluster-id=$CLUSTER_ID \
  -Dkubernetes.container.image.ref=$IMAGE \
  -Dkubernetes.service-account=$FLINK_SERVICE_ACCOUNT \
  local:///opt/flink/examples/streaming/Iteration.jar
```

```

2022-12-29 21:13:06,947 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
    [] - Kubernetes deployment requires a fixed port. Configuration
    blob.server.port will be set to 6124
2022-12-29 21:13:06,948 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
    [] - Kubernetes deployment requires a fixed port. Configuration
    taskmanager.rpc.port will be set to 6122
2022-12-29 21:13:07,861 WARN
    org.apache.flink.kubernetes.KubernetesClusterDescriptor    [] - Please note that
    Flink client operations(e.g. cancel, list, stop, savepoint, etc.) won't work from
    outside the Kubernetes cluster since 'kubernetes.rest-service.exposed.type' has
    been set to ClusterIP.
2022-12-29 21:13:07,868 INFO
    org.apache.flink.kubernetes.KubernetesClusterDescriptor    [] - Create flink
    application cluster flink-application-cluster successfully, JobManager Web
    Interface: http://flink-application-cluster-rest.flink:8081

```

6. Examine los recursos de Kubernetes creados.

```

kubectl get all -n <namespace>
NAME READY STATUS RESTARTS AGE
pod/flink-application-cluster-546687cb47-w2p2z 1/1 Running 0 3m37s
pod/flink-application-cluster-taskmanager-1-1 1/1 Running 0 3m24s

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

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

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

```

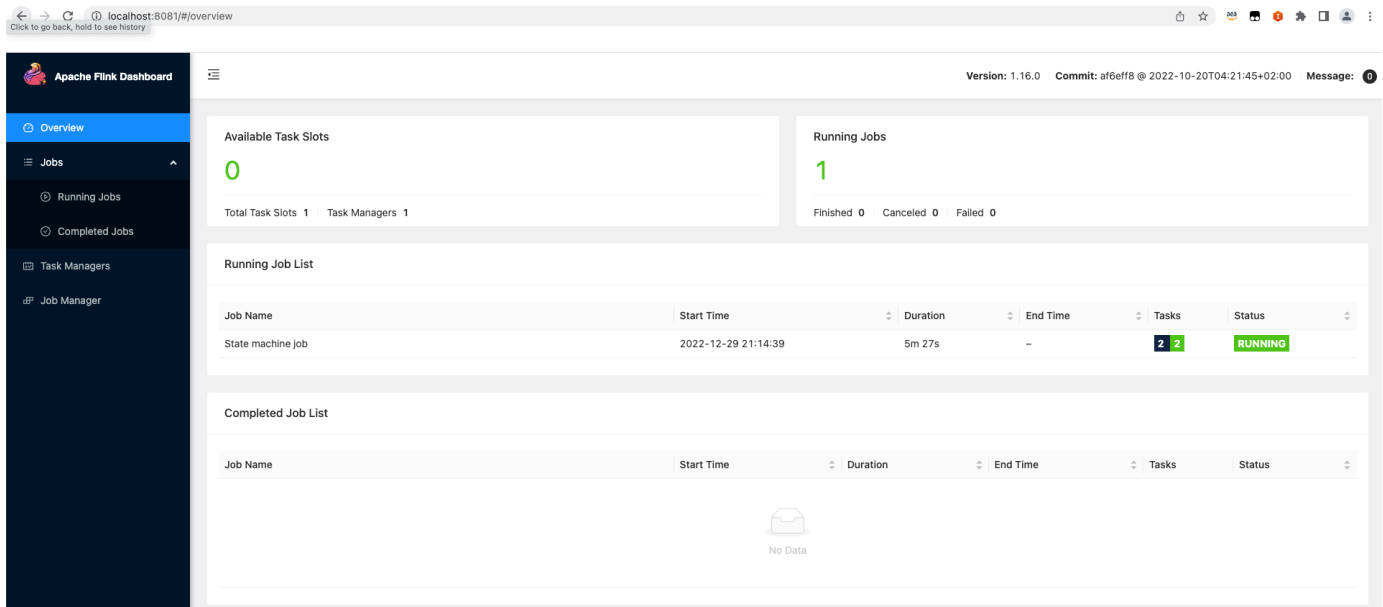
7. Reenvío del puerto a 8081.

```

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

```

8. Acceda localmente a la interfaz de usuario de Flink.



9. Elimine la aplicación de Flink.

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

Para obtener más información sobre cómo enviar solicitudes a Flink, consulte [Kubernetes nativo](#) en la documentación de Apache Flink.

Requisitos de seguridad de la cuenta de JobManager servicio de Flink para Kubernetes nativos

El JobManager pod Flink usa una cuenta de servicio de Kubernetes para acceder al servidor API de Kubernetes y crear y ver los pods. TaskManager La cuenta JobManager de servicio debe tener los permisos adecuados para crear o eliminar TaskManager pods y permitir que el líder ConfigMaps de vigilancia recupere la dirección de tu TaskManager clúster y su ubicación. JobManager ResourceManager

Las siguientes reglas se aplican a esta cuenta de servicio:

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
```

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

```

Personalización de las imágenes de Docker para Flink y FluentD

Siga estos pasos para personalizar las imágenes de Docker de Amazon EMR en EKS con imágenes de Apache Flink o FluentD. Estos incluyen orientación técnica para obtener una imagen base, personalizarla, publicarla y enviar una carga de trabajo.

Temas

- [Requisitos previos](#)
- [Paso 1: recupere una imagen base del Amazon Elastic Container Registry](#)
- [Paso 2: personalizar una imagen base](#)
- [Paso 3: publique una imagen personalizada](#)
- [Paso 4: envíe una carga de trabajo de Flink en Amazon EMR mediante una imagen personalizada](#)

Requisitos previos

Antes de personalizar su imagen Docker, asegúrese de que cumple los siguientes requisitos previos:

- Ha completado los pasos para la [Configuración del operador de Kubernetes de Flink para Amazon EMR en EKS](#).

- Ha instalado Docker en su entorno. Para obtener más información, consulte [Obtener Docker](#).

Paso 1: recupere una imagen base del Amazon Elastic Container Registry

La imagen base contiene el tiempo de ejecución de Amazon EMR y los conectores que necesita para acceder a otros Servicios de AWS. Si utiliza Amazon EMR en EKS con Flink en la versión 6.14.0 o posterior, puede obtener las imágenes base en la Galería pública de Amazon ECR. Navegue por la galería para encontrar el enlace a la imagen y llevarla a su espacio de trabajo local. Por ejemplo, para la versión 6.14.0 de Amazon EMR, el comando `docker pull` le devuelve la imagen base estándar más reciente. Sustituya `emr-6.14.0:latest` por la versión de lanzamiento que desee.

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

Los siguientes son enlaces a la imagen de la galería Flink y a la imagen de la galería Fluentd:

- [emr-on-eks/flink/emr-6.14.0-flink](#)
- [emr-on-eks/fluentd/emr-6.14.0 \(](#)

Paso 2: personalizar una imagen base

Los pasos siguientes describen cómo personalizar la imagen base que ha extraído de Amazon ECR.

1. Cree un Dockerfile nuevo en su espacio de trabajo local.
2. Edite el Dockerfile y añada el siguiente contenido. Este Dockerfile usa la imagen del contenedor que extrajo de `public.ecr.aws/emr-on-eks/flink/emr-7.9.0-flink:latest`.

```
FROM public.ecr.aws/emr-on-eks/flink/emr-7.9.0-flink:latest
USER root
### Add customization commands here ###
USER hadoop:hadoop
```

Utilice la siguiente configuración si está utilizando Fluentd.

```
FROM public.ecr.aws/emr-on-eks/fluentd/emr-7.9.0:latest
USER root
### Add customization commands here ###
```

```
USER hadoop:hadoop
```

3. Agregue comandos en el Dockerfile para personalizar la imagen base. El comando siguiente muestra cómo instalar bibliotecas de Python.

```
FROM public.ecr.aws/emr-on-eks/flink/emr-7.9.0-flink:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. En el mismo directorio en el que se creó DockerFile, ejecute el siguiente comando para crear la imagen de Docker. El campo que proporcione después del indicador de -t es su nombre personalizado para la imagen.

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

Paso 3: publique una imagen personalizada

Ahora puede publicar la nueva imagen de Docker en su registro de Amazon ECR.

1. Ejecute el siguiente comando para crear un repositorio de Amazon ECR para almacenar la imagen de Docker. Proporcione un nombre para su repositorio, por ejemplo `emr_custom_repo`. Para obtener más información, consulte [Creación de un repositorio](#) en la Guía del usuario de Amazon Elastic Container Registry.

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

2. Ejecute el siguiente comando para autenticarse en el registro predeterminado. Para obtener más información, consulte [Autenticación de sus registros personalizados](#) en la Guía del usuario de Amazon Elastic Container Registry.

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

3. Inserte la imagen. Para obtener más información, consulte [Insertar una imagen en Amazon ECR](#) en la Guía del usuario de Amazon Elastic Container Registry.

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

Paso 4: envíe una carga de trabajo de Flink en Amazon EMR mediante una imagen personalizada

Realice los siguientes cambios en sus especificaciones `FlinkDeployment` para usar una imagen personalizada. Para ello, introduzca su propia imagen en la línea `spec.image` de sus especificaciones de implementación.

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

Para usar una imagen personalizada para su trabajo de Fluentd, introduzca su propia imagen en la línea `monitoringConfiguration.image` de sus especificaciones de implementación.

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

Supervisión del operador de Flink Kubernetes y de los trabajos de Flink

En esta sección, se describen varias formas en las que puede supervisar sus trabajos de Flink con Amazon EMR en EKS. Estas incluyen la integración de Flink con el Amazon Managed Service

para Prometheus, el uso del panel web de Flink, que proporciona el estado y las métricas de los trabajos, o el uso de una configuración de monitorización para enviar datos de registro a Amazon S3 y Amazon CloudWatch.

Temas

- [Uso de Amazon Managed Service para Prometheus para supervisar los trabajos de Flink](#)
- [Uso de la IU de Flink para supervisar los trabajos de Flink](#)
- [Uso de la configuración de supervisión para supervisar el operador de Flink Kubernetes y los trabajos de Flink](#)

Uso de Amazon Managed Service para Prometheus para supervisar los trabajos de Flink

Puede integrar Apache Flink a Amazon Managed Service para Prometheus (portal de administración). Amazon Managed Service para Prometheus admite la ingesta de métricas de Amazon Managed Service para servidores de Prometheus en clústeres que se ejecuten en Amazon EKS. Amazon Managed Service para Prometheus funciona junto con un servidor de Prometheus que ya se esté ejecutando en su clúster de Amazon EKS. Al ejecutar la integración de Amazon Managed Service para Prometheus con el operador de Flink de Amazon EMR, un servidor de Prometheus se implementará y configurará automáticamente para que se integre con Amazon Managed Service para Prometheus.

1. [Cree un espacio de trabajo de Amazon Managed Service para Prometheus](#). Este espacio de trabajo sirve como punto de conexión de ingestión. Más adelante necesitará la URL de escritura remota.
2. Configure roles de IAM para cuentas de servicio.

Para este método de incorporación, utilice roles de IAM para las cuentas de servicio del clúster de Amazon EKS en el que se ejecuta el servidor de Prometheus. Estos roles también se denominan roles de servicio.

Si aún no tiene los roles, [configure los roles de servicio para la ingesta de métricas de los clústeres de Amazon EKS](#).

Antes de continuar, cree un rol de IAM llamado `amp-iamproxy-ingest-role`.

3. Instale el operador de Flink de Amazon EMR con Amazon Managed Service para Prometheus.

Ahora que dispone de un espacio de trabajo de Amazon Managed Service para Prometheus, tiene un rol de IAM dedicado para Amazon Managed Service para Prometheus y cuenta con los permisos necesarios, puede instalar el operador de Flink de Amazon EMR.

Cree un archivo `enable-amp.yaml`. Este archivo te permite usar una configuración personalizada para anular la configuración de Amazon Managed Service para Prometheus. Asegúrese de utilizar sus propios roles.

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

Utilice el comando `Helm Install --set` para pasar las anulaciones al gráfico de `flink-kubernetes-operator`.

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

El comando instala automáticamente un informador de Prometheus en el operador del puerto 9999. Cualquier futuro `FlinkDeployment` también expone un puerto `metrics` en 9249.

- Las métricas del operador de Flink aparecen en Prometheus con la etiqueta `flink_k8soperator_`.
- Las métricas de Flink TaskManager aparecen en Prometheus con la etiqueta `flink_taskmanager_`.

- Las métricas de Flink JobManager aparecen en Prometheus con la etiqueta `flink_jobmanager_`.

Uso de la IU de Flink para supervisar los trabajos de Flink

Para supervisar el estado y el rendimiento de una aplicación de Flink en ejecución, utilice el panel web de Flink. Este panel proporciona información sobre el estado del trabajo, el número TaskManagers, las métricas y los registros del trabajo. También le permite ver y modificar la configuración del trabajo de Flink e interactuar con el clúster de Flink al enviar o cancelar trabajos.

Para acceder al panel web de Flink de una aplicación de Flink en ejecución en Kubernetes:

1. Utilice el `kubectl port-forward` comando para reenviar un puerto local al puerto en el que se ejecuta el panel web de Flink en los pods de TaskManager la aplicación Flink. De forma predeterminada, este puerto es 8081. `deployment-name` Sustitúyalo por el nombre de la implementación de la aplicación Flink que aparece arriba.

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

Ejemplo de salida:

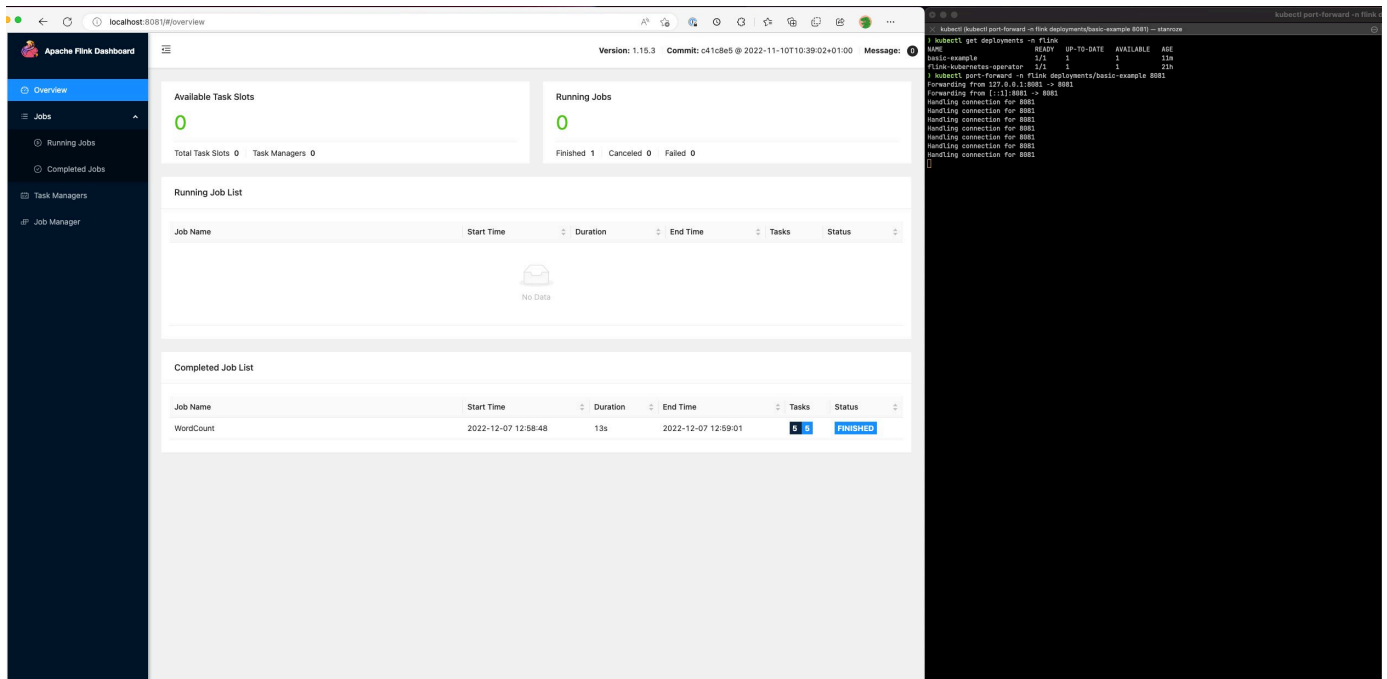
```
kubectl get deployments -n flink-namespace
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
basic-example                      1/1      1              1            11m
flink-kubernetes-operator          1/1      1              1            21h
```

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

2. Si desea utilizar un puerto diferente de forma local, utilice el parámetro: `8081` `local-port`.

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

3. En un navegador web, vaya a `http://localhost:8081` (o `http://localhost:local-port` si utilizó un puerto local personalizado) para acceder al panel web de Flink. Este panel muestra información sobre la aplicación Flink en ejecución, como el estado del trabajo, el número y las métricas y los registros del TaskManagers trabajo.



Uso de la configuración de supervisión para supervisar el operador de Flink Kubernetes y los trabajos de Flink

La configuración de monitoreo le permite configurar fácilmente el archivado de registros de su aplicación Flink y los registros del operador en S3 y/o CloudWatch (puede elegir uno o ambos). Al hacerlo, se añade un sidecar FluentD a los módulos TaskManager y, posteriormente, se JobManager reenvían los registros de estos componentes a los sumideros configurados.

Note

Debe configurar los roles de IAM para la cuenta de servicio de su operador de Flink y su trabajo de Flink (cuentas de servicio) para poder utilizar esta característica, ya que requiere interactuar con otros Servicios de AWS. Debe configurarlo con las IRSA en [Configuración del operador de Kubernetes de Flink para Amazon EMR en EKS](#).

Registros de aplicaciones de Flink

Puede definir la configuración de la siguiente manera:

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

```

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

```

Las siguientes son opciones de configuración.

- `s3MonitoringConfiguration`: clave de configuración para configurar el reenvío a S3
- `logUri` (obligatorio): la ruta del bucket de S3 donde desea almacenar sus registros.
- Una vez cargados los registros, la ruta en S3 tendrá el siguiente aspecto.
- La rotación de los registros no está habilitada:

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


- La rotación de los registros está habilitada. Puede utilizar tanto un archivo rotado como un archivo actual (uno sin la fecha).

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

El siguiente formato es un número creciente.

```
s3://${logUri}/${POD_NAME}/stdout_YYYYMMDD_index.gz
```

- Los siguientes permisos de IAM son necesarios para utilizar este reenviador.

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

- `cloudWatchMonitoringConfiguration`— clave de configuración para configurar el reenvío. CloudWatch
- `logGroupName`(obligatorio): nombre del grupo de CloudWatch registros al que desea enviar los registros (crea automáticamente el grupo si no existe).
- `logStreamNamePrefix` (opcional): nombre del flujo de registro al que quiere enviar registros. El valor predeterminado es una cadena vacía. El formato es el siguiente:

```
${logStreamNamePrefix}/${POD_NAME}/STDOUT or STDERR
```

- Los siguientes permisos de IAM son necesarios para utilizar este reenviador.

```
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents"
  ],
}
```

```

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

```

- **sideCarResources** (opcional): la clave de configuración para establecer los límites de recursos en el contenedor asociado de Fluentbit lanzado.
- **memoryLimit** (opcional): el valor predeterminado es 512 Mi. Ajústelo según sus necesidades.
- **cpuLimit** (opcional): esta opción no tiene un valor predeterminado. Ajústelo según sus necesidades.
- **containerLogRotationConfiguration** (opcional): controla el comportamiento de la rotación del registro de contenedor. Está habilitada de forma predeterminada.
 - **rotationSize** (obligatorio): especifica el tamaño del archivo para la rotación del registro. El rango de valores posibles va de 2 KB a 2 GB. La parte de la unidad numérica del parámetro **rotationSize** se pasa como un número entero. Como no se admiten valores decimales, puede especificar un tamaño de rotación de 1,5 GB, por ejemplo, con el valor 1500 MB. El valor predeterminado es 2 GB.
 - **maxFilesToKeep** (obligatorio): especifica el número máximo de archivos que deben retenerse en el contenedor una vez hecha la rotación. El valor mínimo es 1 y el máximo, 50. El valor predeterminado es 10.

Registros del operador de Flink

También podemos habilitar el archivado de registros para el operador al utilizar las siguientes opciones en el archivo `values.yaml` en la instalación del gráfico de Helm. Puede activar S3 o ambas opciones. CloudWatch

```

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

```

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

A continuación se describen las opciones de configuración disponibles de `monitoringConfiguration`.

- `s3MonitoringConfiguration`: configure esta opción para archivar en S3.
- `logUri` (obligatorio): la ruta del bucket de S3 donde desea almacenar sus registros.
- Los siguientes son formatos de cómo se verían las rutas del bucket de S3 una vez cargados los registros.
- La rotación de los registros no está habilitada.

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

- La rotación de los registros está habilitada. Puede utilizar tanto un archivo rotado como un archivo actual (uno sin la fecha).

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

El siguiente índice de formato es un número creciente.

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

- `cloudWatchMonitoringConfiguration`— la clave de configuración a la que se debe configurar el reenvío. CloudWatch
- `logGroupName` (obligatorio): nombre del grupo de CloudWatch registros al que desea enviar los registros. Si el grupo no existe, se crea automáticamente.
- `logStreamNamePrefix` (opcional): nombre del flujo de registro al que quiere enviar registros. El valor predeterminado es una cadena vacía. El formato CloudWatch es el siguiente:

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

- `sideCarResources` (opcional): la clave de configuración para establecer los límites de recursos en el contenedor asociado de Fluentbit lanzado.
- `memoryLimit` (opcional): el límite de memoria. Ajústelo según sus necesidades. El valor predeterminado es 512Mi.

- `cpuLimit`: el límite de la CPU. Ajústelo según sus necesidades. Sin valor predeterminado.
- `containerLogRotationConfiguration` (opcional): controla el comportamiento de la rotación de los registro de contenedor. Está habilitada de forma predeterminada.
- `rotationSize` (obligatorio): especifica el tamaño del archivo para la rotación del registro. El rango de valores posibles va de 2 KB a 2 GB. La parte de la unidad numérica del parámetro `rotationSize` se pasa como un número entero. Como no se admiten valores decimales, puede especificar un tamaño de rotación de 1,5 GB, por ejemplo, con el valor 1500 MB. El valor predeterminado es 2 GB.
- `maxFilesToKeep` (obligatorio): especifica el número máximo de archivos que deben retenerse en el contenedor una vez hecha la rotación. El valor mínimo es 1 y el máximo, 50. El valor predeterminado es 10.

Cómo apoya Flink la alta disponibilidad y la resiliencia al trabajo

En las siguientes secciones, se explica cómo Flink hace que los trabajos sean más fiables y tengan una alta disponibilidad. Lo hace a través de capacidades integradas, como la alta disponibilidad de Flink y varias capacidades de recuperación en caso de que se produzcan fallos.

Temas

- [Uso de la alta disponibilidad \(HA\) para Flink Operators y Flink Applications](#)
- [Optimización de los tiempos de reinicio de las tareas de Flink para las operaciones de escalado y de recuperación de tareas con Amazon EMR en EKS](#)
- [Desactivación rápida de las instancias de spot con Flink en Amazon EMR en EKS](#)

Uso de la alta disponibilidad (HA) para Flink Operators y Flink Applications

En este tema se muestra cómo configurar la alta disponibilidad y se describe cómo funciona en algunos casos de uso diferentes. Aquí se incluye cuando usa el administrador de trabajos y cuando usa Kubernetes nativo de Flink.

Alta disponibilidad del Flink Operator

Habilitamos la alta disponibilidad para el Flink Operator para poder realizar un cambio a un Flink Operator en espera y minimizar el tiempo de inactividad en el bucle de control del operador en caso de fallos. La alta disponibilidad está habilitada de forma predeterminada y el número predeterminado

de réplicas de operadores de inicio es 2. Puede configurar el campo de réplicas en su archivo `values.yaml` para el gráfico de Helm.

Los siguientes campos se pueden personalizar:

- `replicas` (opcional, el valor predeterminado es 2): si se establece este número en uno mayor que 1, se crean otros operadores en espera y se puede recuperar el trabajo con mayor rapidez.
- `highAvailabilityEnabled` (opcional, el valor predeterminado es verdadero): controla si desea habilitar la alta disponibilidad. Si se especifica este parámetro como verdadero, se habilita la compatibilidad con la implementación multi-AZ y se establecen los parámetros `flink-conf.yaml` correctos.

Puede deshabilitar la alta disponibilidad para su operador al establecer la siguiente configuración en su archivo `values.yaml`.

```
...
imagePullSecrets: []

replicas: 1

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

Implementación Multi-AZ

Creamos los pods de los operadores en varias zonas de disponibilidad. Se trata de una limitación leve y, si no dispone de recursos suficientes en una zona de disponibilidad diferente, los pods de los operadores se programarán en la misma zona de disponibilidad.

Determinar la réplica líder

Si HA está habilitada, las réplicas utilizan un arrendamiento para determinar cuál de ellos JM es el líder y utilizan un arrendamiento de K8s para la elección del líder. Puede describir el arrendamiento y consultar el campo `.Spec.Holder Identity` para determinar el líder actual

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

Interacción entre Flink y S3

Configuración de las credenciales de acceso

Asegúrese de haber configurado las IRSA con los permisos de IAM adecuados para acceder al bucket de S3.

Búsqueda de archivos jar de trabajo desde el modo aplicación de S3

El operador de Flink también admite la búsqueda de archivos jar de aplicaciones desde S3. Solo tiene que proporcionar la ubicación S3 del JaRuri en su especificación. `FlinkDeployment`

También puedes usar esta función para descargar otros artefactos, como PyFlink scripts. El script de Python resultante se coloca en la ruta `/opt/flink/usrlib/`.

El siguiente ejemplo muestra cómo utilizar esta función para un PyFlink trabajo. Tenga en cuenta los campos `jarURI` y `args`.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  image: <YOUR CUSTOM PYFLINK IMAGE>
  emrReleaseLabel: "emr-6.12.0-flink-latest"
  flinkVersion: v1_16
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  serviceAccount: flink
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: "s3://<S3-BUCKET>/scripts/pyflink.py" # Note, this will trigger the
    artifact download process
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-pyclientexec", "/usr/local/bin/python3", "-py", "/opt/flink/usrlib/
    pyflink.py"]
```

```
parallelism: 1
upgradeMode: stateless
```

Conectores de Flink de S3

Flink viene empaquetado con dos conectores de S3 (enumerados a continuación). En las siguientes secciones se explica cuándo usar cada conector.

Punto de control: conector de Presto de S3

- Establezca el esquema de S3 en `s3p://`
- El conector recomendado para establecer el punto de control en s3. Para obtener más información, consulte [Específico de S3](#) en la documentación de Apache Flink.

Ejemplo de FlinkDeployment especificación:

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

Lectura y escritura en S3: conector Hadoop S3

- Establezca el esquema de S3 en `s3://` o en (`s3a://`)
- El conector recomendado para leer y escribir archivos desde S3 (solo el conector de S3 que implementa la interfaz del [sistema de archivos de Flink](#)).
- De forma predeterminada, establecemos `fs.s3a.aws.credentials.provider` en el archivo `flink-conf.yaml`, que es `com.amazonaws.auth.WebIdentityTokenCredentialsProvider`. Si anula completamente el valor predeterminado `flink-conf` y está interactuando con S3, asegúrese de usar este proveedor.

Ejemplo de FlinkDeployment especificación

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

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

Flink JobManager

La alta disponibilidad (HA) de las implementaciones de Flink permite que los trabajos sigan progresando incluso si se produce un error transitorio y el usuario se bloquea. Los trabajos se reiniciarán, pero desde el último punto de control exitoso con la alta disponibilidad activada. Si la HA no está habilitada, Kubernetes lo reiniciará JobManager, pero su trabajo empezará como uno nuevo y perderá el progreso. Tras configurar HA, podemos indicarle a Kubernetes que almacene los metadatos de alta disponibilidad en un almacenamiento persistente como referencia en caso de que se produzca un fallo transitorio JobManager y que, después, reanude nuestras tareas desde el último punto de control exitoso.

La alta disponibilidad está habilitada de forma predeterminada para sus trabajos de Flink (el recuento de réplicas está establecido en 2, por lo que tendrá que proporcionar una ubicación de almacenamiento en S3 para que los metadatos de alta disponibilidad se conserven).

Configuraciones de alta disponibilidad

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    executionRoleArn: "<JOB_EXECUTION_ROLE_ARN>"
    emrReleaseLabel: "emr-6.13.0-flink-latest"
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    replicas: 2
```



```
highAvailabilityEnabled: true
storageDir: "s3://<S3 PERSISTENT STORAGE DIR>"
taskManager:
  resource:
    memory: "2048m"
    cpu: 1
```

Las siguientes son descripciones de las configuraciones de alta disponibilidad anteriores en Job Manager (definidas en `.spec.JobManager`):

- `highAvailabilityEnabled` (opcional, el valor predeterminado es verdadero): configúrela en `false` si no quiere activar la alta disponibilidad y no quiere utilizar las configuraciones de alta disponibilidad proporcionadas. Aún puede manipular el campo “réplicas” para configurar de forma manual la alta disponibilidad.
- `replicas` (opcional, el valor predeterminado es 2): si se establece este número en mayor que 1, se crea otro modo de espera JobManagers y permite una recuperación más rápida del trabajo. Si deshabilita la alta disponibilidad, debe establecer el recuento de réplicas en 1 o seguirá recibiendo errores de validación (solo se admite 1 réplica si la alta disponibilidad no está habilitada).
- `storageDir` (obligatorio): dado que utilizamos el recuento de réplicas como 2 de forma predeterminada, debemos proporcionar un `storageDir` persistente. Actualmente, este campo solo acepta rutas de S3 como ubicación de almacenamiento.

Localidad del pod

Si habilitas la alta disponibilidad, también intentamos colocar los pods en la misma zona de disponibilidad, lo que mejora el rendimiento (se reduce la latencia de la red al tener los pods en la misma zona AZs). Se trata de un proceso de mejor esfuerzo, ya que, si no dispone de recursos suficientes en la zona de disponibilidad en la que están programados la mayoría de sus pods, los demás pods seguirán estando programados, pero es posible que acaben en un nodo fuera de esta zona de disponibilidad.

Determinar la réplica líder

Si la alta disponibilidad está habilitada, las réplicas utilizan un arrendamiento para determinar cuál de ellos JM es el líder y utilizan un mapa de configuración del K8s como almacén de datos para almacenar estos metadatos. Si quiere determinar el líder, consulte el contenido del Configmap y busque en los datos la clave `org.apache.flink.k8s.leader.restserver` para encontrar los pod de K8 con la dirección IP. También puede utilizar los siguientes comandos bash.

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

Trabajo de Flink: Kubernetes nativo

A partir de la versión 6.13.0, Amazon EMR es compatible con Kubernetes nativo de Flink para ejecutar aplicaciones de Flink en un clúster de Amazon EKS.

Note

Debe tener un bucket de Amazon S3 creado para almacenar los metadatos de alta disponibilidad cuando envíe su trabajo de Flink. Si no desea usar esta característica, puede desactivarla. Está habilitada de forma predeterminada.

Para activar la característica de alta disponibilidad de Flink, utilice los siguientes parámetros de Flink al [ejecutar el comando de la CLI run-application](#). Los parámetros se definen debajo del ejemplo.

```
-Dhigh-availability.type=kubernetes \
-Dhigh-availability.storageDir=S3://DOC-EXAMPLE-STORAGE-BUCKET \
-
Dfs.s3a.aws.credentials.provider="com.amazonaws.auth.WebIdentityTokenCredentialsProvider" \
-Dkubernetes.jobmanager.replicas=3 \
-Dkubernetes.cluster-id=example-cluster
```

- **Dhigh-availability.storageDir**: el bucket de Amazon S3 en el que desea almacenar los metadatos de alta disponibilidad para su trabajo.

Dkubernetes.jobmanager.replicas: el número de pods de Job Manager que se van a crear como un entero superior a 1.

Dkubernetes.cluster-id: un identificador único que identifica el clúster de Flink.

Optimización de los tiempos de reinicio de las tareas de Flink para las operaciones de escalado y de recuperación de tareas con Amazon EMR en EKS

Cuando se produce un error en una tarea o se produce una operación de escalado, Flink intenta volver a ejecutar la tarea desde el último punto de control completado. El proceso de reinicio puede tardar un minuto o más en ejecutarse, según el tamaño del estado del punto de control y el número de tareas paralelas. Durante el periodo de reinicio, es posible que se acumulen tareas pendientes para el trabajo. Sin embargo, hay algunas formas en las que Flink optimiza la velocidad de los gráficos de recuperación y reinicio de la ejecución para mejorar la estabilidad del trabajo.

En esta página, se describen algunas de las formas en que Amazon EMR Flink puede mejorar el tiempo de reinicio de un trabajo durante las operaciones de recuperación o escalado en instancias Spot. Las instancias Spot representan capacidad informática no utilizada que está disponible con descuento. Tiene comportamientos únicos, incluidas las interrupciones ocasionales, por lo que es importante entender cómo Amazon EMR en EKS los gestiona, incluida la forma en que Amazon EMR en EKS lleva a cabo el desmantelamiento y el reinicio de los trabajos.

Temas

- [Recuperación local de tareas](#)
- [Recuperación local de tareas mediante el montaje de volúmenes de Amazon EBS](#)
- [Punto de control incremental genérico basado en registros](#)
- [Recuperación detallada](#)
- [Mecanismo de reinicio combinado en el programador adaptativo](#)

Recuperación local de tareas

Note

La recuperación local de tareas es compatible a partir de la versión 6.14.0 de Flink en Amazon EMR en EKS.

Con los puntos de control de Flink, cada tarea produce una instantánea de su estado que Flink graba en un almacenamiento distribuido como Amazon S3. En los casos de recuperación, las tareas restauran su estado desde el almacenamiento distribuido. El almacenamiento distribuido ofrece

tolerancia a errores y puede redistribuir el estado durante el reescalado, ya que todos los nodos pueden acceder a él.

Sin embargo, un almacén distribuido remoto también tiene una desventaja: todas las tareas deben leer su estado desde una ubicación remota a través de la red. Esto puede provocar tiempos de recuperación prolongados para los estados de gran tamaño durante las operaciones de recuperación o escalado de tareas.

Este problema del tiempo de recuperación prolongado se resuelve mediante la recuperación local de tareas. Las tareas escriben su estado en el punto de control de un almacenamiento secundario que es local para la tarea, por ejemplo, en un disco local. También almacenan su estado en el almacenamiento principal, o Amazon S3 en nuestro caso. Durante la recuperación, el programador programa las tareas en el mismo administrador de tareas en el que se ejecutaron anteriormente, de modo que se puedan recuperar del almacén de estado local en lugar de leerlas del almacén de estado remoto. Para obtener más información, consulte [Task-Local Recovery](#) en la documentación de Apache Flink.

Nuestras pruebas comparativas con trabajos de muestra han demostrado que el tiempo de recuperación se ha reducido de unos minutos a unos pocos segundos con la opción de recuperación local de tareas habilitada.

Para habilitar la recuperación local de tareas, defina las siguientes configuraciones en el archivo `flink-conf.yaml`. Especifique el valor del intervalo de puntos de control en milisegundos.

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

Recuperación local de tareas mediante el montaje de volúmenes de Amazon EBS

Note

La recuperación local de tareas mediante Amazon EBS es compatible a partir de la versión 6.15.0 de Flink en Amazon EMR en EKS.

Con Flink en Amazon EMR en EKS, puede aprovisionar automáticamente volúmenes de Amazon EBS al TaskManager pods para la recuperación local de tareas. El soporte superpuesto

predeterminado viene con un volumen de 10 GB, suficiente para trabajos con un estado inferior. Los trabajos con estados grandes pueden habilitar la opción de montaje de volumen automático con EBS. La TaskManager Los pods se crean y montan automáticamente durante la creación del pod y se eliminan al eliminarlos.

Realice los siguientes pasos a fin de habilitar el montaje automático del volumen de EBS para Flink en Amazon EMR en EKS:

1. Exporte los valores de las siguientes variables que utilizará en los próximos pasos.

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

2. Cree o actualice un archivo de kubeconfig de YAML para el clúster.

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

3. Cree una cuenta de servicio de IAM para el controlador de Amazon EBS Container Storage Interface (CSI) en su clúster de Amazon EKS.

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

4. Cree un controlador del Amazon EBS CSI con el siguiente comando:

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

5. Cree la clase de almacenamiento de Amazon EBS con el siguiente comando:

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

Y, a continuación, aplique la clase:

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

6. Helm instaló el operador Amazon EMR Flink Kubernetes con opciones para crear una cuenta de servicio. Esto crea el `emr-containers-sa-flink` que se usará en la implementación de Flink.

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

7. Para enviar el trabajo de Flink y habilitar el aprovisionamiento automático de volúmenes de EBS para la recuperación local de tareas, defina las siguientes configuraciones en el archivo `flink-conf.yaml`. Ajuste el límite de tamaño para el tamaño del estado del trabajo. Establece `serviceAccount` en `emr-containers-sa-flink`. Especifique el valor del intervalo de puntos de control en milisegundos. Y omita el `executionRoleArn`.

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

Cuando esté listo para eliminar el complemento del controlador CSI de Amazon EBS, utilice los siguientes comandos:

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

Punto de control incremental genérico basado en registros

Note

Los puntos de control incrementales genéricos basados en registros son compatibles a partir de la versión 6.14.0 de Flink en Amazon EMR en EKS.

En Flink 1.16, se agregaron puntos de control incrementales genéricos basados en registros para mejorar la velocidad de los puntos de control. Un intervalo de puntos de control más rápido suele reducir el trabajo de recuperación, ya que es necesario volver a procesar menos eventos después de la recuperación. Para obtener más información, consulte [Improving speed and stability of checkpointing with generic log-based incremental checkpoints](#) en el blog de Apache Flink.

Con trabajos de muestra, nuestras pruebas comparativas han demostrado que el tiempo de los puntos de control se ha reducido de unos minutos a unos pocos segundos con el punto de control incremental genérico basado en registros.

Para habilitar los puntos de control incrementales genéricos basados en registros, defina las siguientes configuraciones en el archivo `flink-conf.yaml`. Especifique el valor del intervalo de puntos de control en milisegundos.

```
state.backend.changelog.enabled: true
```

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

Recuperación detallada

Note

El soporte de la recuperación detallada para el programador predeterminado es compatible a partir de la versión 6.14.0 de Flink en Amazon EMR en EKS. El soporte de la recuperación detallada para el programador predeterminado está disponible a partir de la versión 6.15.0 de Flink en Amazon EMR en EKS.

Cuando se produce un error en una tarea durante la ejecución, Flink restablece todo el gráfico de ejecución y activa una nueva ejecución completa desde el último punto de control completado. Esto es más caro que volver a ejecutar las tareas con errores. La recuperación detallada reinicia solo el componente conectado a la canalización de la tarea con el error. En el siguiente ejemplo, el gráfico de tareas tiene 5 vértices (de A a E). Todas las conexiones entre los vértices se canalizan con una distribución puntual y el valor `parallelism.default` para el trabajo se establece en 2.

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

En este ejemplo, hay un total de 10 tareas en ejecución. La primera canalización (a1toe1) se ejecuta en un TaskManager (TM1), y la segunda canalización (a2haciae2) corre por otra TaskManager (TM2).

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

Hay dos componentes conectados por canalización: a1 # e1 y a2 # e2. Si TM1 alguna de las dos TM2 falla, la falla solo afecta a las 5 tareas en proceso, donde TaskManager estaba ejecutándose. La estrategia de reinicio solo inicia el componente canalizado afectado.

La recuperación detallada solo funciona con trabajos de Flink perfectamente paralelos. No es compatible con las operaciones `keyBy()` o `redistribute()`. Para obtener más información,

consulte [FLIP-1: Fine Grained Recovery from Task Failures](#) en el proyecto de Jira Flink Improvement Proposal.

Para habilitar la recuperación detallada, establezca las siguientes configuraciones en el archivo `flink-conf.yaml`.

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

Mecanismo de reinicio combinado en el programador adaptativo

Note

El mecanismo de reinicio combinado del programador adaptativo es compatible a partir de la versión 6.15.0 de Flink en Amazon EMR en EKS.

El programador adaptativo puede ajustar el paralelismo del trabajo en función de los espacios disponibles. Reduce automáticamente el paralelismo si no hay suficientes espacios disponibles para ajustarse al paralelismo del trabajo configurado. Si hay nuevos espacios disponibles, la tarea se escala verticalmente de nuevo al paralelismo del trabajo configurado. Un programador adaptativo evita el tiempo de inactividad del trabajo cuando no hay suficientes recursos disponibles. Este es el programador compatible con el escalador automático de Flink. Recomendamos el programador adaptativo con Amazon EMR Flink por estos motivos. Sin embargo, los programadores adaptativos pueden realizar varios reinicios en un periodo corto, uno por cada nuevo recurso agregado. Esto puede provocar una disminución en el rendimiento del trabajo.

Con Amazon EMR 6.15.0 y versiones posteriores, Flink cuenta con un mecanismo de reinicio combinado en el programador adaptativo que abre una ventana de reinicio cuando se agrega el primer recurso y, a continuación, espera hasta alcanzar el intervalo de ventana configurado del minuto predeterminado. Realiza un único reinicio cuando hay suficientes recursos disponibles para ejecutar el trabajo con el paralelismo configurado o cuando se agota el intervalo.

Con trabajos de muestra, nuestras pruebas comparativas han demostrado que esta característica procesa un 10 % de los registros más que el comportamiento predeterminado cuando se utiliza el programador adaptativo y el escalador automático de Flink.

Para habilitar el mecanismo de reinicio combinado, defina las siguientes configuraciones en el archivo `flink-conf.yaml`.

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

Desactivación rápida de las instancias de spot con Flink en Amazon EMR en EKS

Flink con Amazon EMR en EKS puede mejorar el tiempo de reinicio de un trabajo durante las operaciones de recuperación o escalado de tareas.

Descripción general

A partir de las versiones 6.15.0, Amazon EMR en EKS es compatible con la desactivación rápida de los administradores de tareas en instancias de sport de Amazon EMR en EKS con Apache Flink. Como parte de esta característica, Amazon EMR en EKS con Flink ofrece las siguientes funciones:

- **Just-in-time puntos de control:** los trabajos de streaming de Flink pueden responder a la interrupción de una instancia puntual, realizar un control just-in-time (JIT) de los trabajos en ejecución e impedir la programación de tareas adicionales en estas instancias puntuales. El punto de control JIT es compatible con el programador predeterminado y adaptativo.
- **Mecanismo de reinicio combinado :** un mecanismo de reinicio combinado hace todo lo posible por reiniciar el trabajo una vez alcanzado el paralelismo de los recursos objetivo o al final de la ventana configurada actualmente. Esto también evita que los trabajos se reinicien de forma consecutiva, lo que podría deberse a la finalización de varias instancias de sport. El mecanismo de reinicio combinado solo está disponible con el programador adaptativo.

Estas capacidades brindan los siguientes beneficios:

- Puede aprovechar las instancias de spot para ejecutar administradores de tareas y reducir el gasto en clústeres.
- La mejora de la agilidad del administrador de tareas de las instancias de spot se traduce en una mayor resiliencia y en una programación de trabajos más eficiente.
- Sus trabajos de Flink tendrán más tiempo de actividad porque se reiniciarán menos tras la finalización de una instancia de Spot.

Cómo funciona una desactivación rápida

Considere el siguiente ejemplo: aprovisiona un clúster de Amazon EMR en EKS que ejecuta Apache Flink y especifica los nodos bajo demanda para el administrador de trabajos y los nodos de las instancias de spot para el administrador de tareas. Dos minutos antes de la finalización, el administrador de tareas recibe un aviso de interrupción.

En este escenario, el administrador de trabajos gestionaría la señal de interrupción de la instancia de spot, bloquearía la programación de tareas adicionales en la instancia de spot e iniciaría los puntos de control JIT para el trabajo de transmisión.

A continuación, el administrador de trabajos reiniciaría el gráfico de tareas solo cuando haya suficiente disponibilidad de nuevos recursos para satisfacer el paralelismo de tareas actual en la ventana de intervalos de reinicio actual. El intervalo de reinicios se decide en función de la duración del reemplazo de la instancia de spot, la creación de nuevos pods del administrador de tareas y el registro en el administrador de trabajos.

Requisitos previos

Para utilizar una desactivación rápida, cree y ejecute un trabajo de streaming en un clúster de Amazon EMR en EKS que ejecute Apache Flink. Habilite el programador adaptativo y los administradores de tareas programados en al menos una instancia de spot, como se muestra en el siguiente ejemplo. Debe utilizar nodos bajo demanda para el administrador de trabajos y puede usar nodos bajo demanda para los administradores de tareas siempre que también haya al menos una instancia de spot.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: deployment_name
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    cluster.taskmanager.graceful-decommission.enabled: "true"
    execution.checkpointing.interval: "240s"
    jobmanager.adaptive-scheduler.combined-restart.enabled: "true"
    jobmanager.adaptive-scheduler.combined-restart.window-interval : "1m"
  serviceAccount: flink
  jobManager:
```

```

resource:
  memory: "2048m"
  cpu: 1
nodeSelector:
  'eks.amazonaws.com/capacityType': 'ON_DEMAND'
taskManager:
  resource:
    memory: "2048m"
    cpu: 1
  nodeSelector:
    'eks.amazonaws.com/capacityType': 'SPOT'
job:
  jarURI: flink_job_jar_path

```

Configuración

En esta sección, se describe la mayoría de las configuraciones que puede especificar para sus necesidades de desactivación.

Clave	Descripción	Valor predeterminado	Valores aceptables
<code>cluster.taskmanager.graceful-decommission.enabled</code>	Habilite la desactivación rápida del administrador de tareas.	true	true, false
<code>jobmanager.adaptive-scheduler.combined-restart.enabled</code>	Habilite el mecanismo de reinicio combinado en el programador adaptativo.	false	true, false
<code>jobmanager.adaptive-scheduler.combined</code>	El intervalo combinado de la ventana de reinicios para realizar los reinicios combinados del trabajo. Un entero	1m	Ejemplos: 30, 60s, 3m, 1h

Clave	Descripción	Valor predeterminado	Valores aceptables
ed-restart.window-interval	sin unidad se interpreta como milisegundos.		

Uso del escalador automático para aplicaciones de Flink

El escalador automático del operador puede ayudar a reducir la contrapresión mediante la recopilación de métricas de los trabajos de Flink y el ajuste automático del paralelismo a nivel de vértice de los trabajos. A continuación se muestra un ejemplo de cómo podría ser su configuración:

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

Esta configuración usa valores predeterminados para la última versión de Amazon EMR. Si utiliza otras versiones, es posible que tengan valores diferentes.

Note

A partir de Amazon EMR 7.2.0, no necesita incluir el prefijo `kubernetes.operator` en la configuración. Si utiliza la versión 7.1.0 o una versión anterior, debe

usar el prefijo antes de cada configuración. Por ejemplo, debe especificar `kubernetes.operator.job.autoscaler.scaling.enabled`.

A continuación se indican las opciones de configuración del escalado automático.

- `job.autoscaler.scaling.enabled`: especifica si el escalador automático debe habilitar la ejecución del escalado de vértices. El valor predeterminado es `true`. Si deshabilita esta configuración, el escalador automático solo recopila métricas y evalúa el paralelismo sugerido para cada vértice, pero no actualiza los trabajos.
- `job.autoscaler.stabilization.interval`: el periodo de estabilización en el que no se ejecutará ningún nuevo escalado. El tiempo predeterminado es 5 minutos.
- `job.autoscaler.metrics.window`: el tamaño de la ventana de agregación de métricas de escalado. Cuanto más grande sea la ventana, mayor será la suavidad y la estabilidad, pero el escalador automático puede ser más lento a la hora de reaccionar ante los cambios repentinos de carga. El tiempo predeterminado es 15 minutos. Le recomendamos que experimente con un valor de entre 3 y 60 minutos.
- `job.autoscaler.target.utilization`: el uso del vértice de destino para proporcionar un rendimiento estable del trabajo y algo de búfer para las fluctuaciones de carga. El valor predeterminado es `0.7`, cuyo objetivo es una utilización o carga del 70 % para los vértices del trabajo.
- `job.autoscaler.target.utilization.boundary`: el límite de uso del vértice de destino que sirve como búfer adicional para evitar el escalado inmediato en caso de fluctuaciones de carga. El valor predeterminado es `0.3`, lo que significa que se permite una desviación del 30 % con respecto a la utilización objetivo antes de activar una acción de escalado.
- `ob.autoscaler.restart.time`: el tiempo previsto para reiniciar la aplicación. El tiempo predeterminado es 5 minutos.
- `job.autoscaler.catch-up.duration`: el tiempo previsto para la recuperación, es decir, procesar por completo cualquier atraso una vez finalizada la operación de escalado. El tiempo predeterminado es 5 minutos. Al reducir la duración de la recuperación, el escalador automático debe reservar más capacidad adicional para las acciones de escalado.
- `pipeline.max-parallelism`: el paralelismo máximo que puede utilizar el escalador automático. El escalador automático ignora este límite si es superior al paralelismo máximo configurado en la configuración de Flink o directamente en cada operador. El valor predeterminado es `-1`. Tenga en cuenta que el escalador automático calcula el paralelismo como un divisor del

número máximo de paralelismo, por lo que se recomienda elegir una configuración de paralelismo máximo que tenga muchos divisores en lugar de confiar en los valores predeterminados proporcionados por Flink. Recomendamos utilizar múltiplos de 60 para esta configuración, como 120, 180, 240, 360, 720, etc.

Para ver una página de referencia de configuración más detallada, consulte [Configuración del escalador automático](#).

Ajuste automático de los parámetros del escalador automático

En esta sección se describe el comportamiento del ajuste automático de varias versiones de Amazon EMR. También trata detalles sobre las diferentes configuraciones de escalado automático.

Note

Amazon EMR 7.2.0 y versiones posteriores utilizan la configuración de código abierto `job.autoscaler.restart.time-tracking.enabled` para permitir la estimación del tiempo de reescalado. La estimación del tiempo de reescalado tiene la misma funcionalidad que el ajuste automático de Amazon EMR, por lo que no es necesario asignar manualmente valores empíricos al tiempo de reinicio.

Puede seguir utilizando el escalado automático de Amazon EMR, si utiliza las versiones 7.1.0 o anteriores de Amazon EMR.

7.2.0 and higher

Amazon EMR 7.2.0 y versiones posteriores miden el tiempo real de reinicio necesario para aplicar las decisiones de escalado automático. En las versiones 7.1.0 y anteriores, tenía que usar la configuración `job.autoscaler.restart.time` para configurar manualmente el tiempo máximo de reinicio estimado. Al usar la configuración `job.autoscaler.restart.time-tracking.enabled`, solo necesita introducir una hora de reinicio para el primer escalado. Después, el operador registra el tiempo de reinicio real y lo usará para los escalados posteriores.

Para habilitar este seguimiento, utilice el siguiente comando:

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

Las siguientes son las configuraciones relacionadas para la estimación del tiempo de reescalado.

Configuración	Obligatorio	Predeterminado/a	Descripción
<code>job.autoscaler.restart.time-tracking.enabled</code>	No	False	Indica si el escalador automático de Flink debe ajustar automáticamente las configuraciones a lo largo del tiempo para optimizar las decisiones de escalado. Tenga en cuenta que el escalador automático solo puede ajustar automáticamente el parámetro <code>restart.time</code> del escalador automático.
<code>job.autoscaler.restart.time</code>	No	5 m	El tiempo de reinicio esperado que utiliza Amazon EMR en EKS hasta que el operador pueda determinar el tiempo de reinicio real a partir de los escalados anteriores.
<code>job.autoscaler.restart.time-tracking.enabled</code>	No	15 m	El tiempo máximo de reinicio observado cuando <code>job.autoscaler.restart.time-tracking.enabled</code> está establecido en <code>true</code> .

El siguiente es un ejemplo de especificación de implementación que puede utilizar para probar la estimación del tiempo de reescalado:

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



```
spec:
  flinkVersion: v1_18
  flinkConfiguration:

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

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

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

  executionRoleArn: <JOB_ARN>
  emrReleaseLabel: emr-7.9.0-flink-latest
  jobManager:
    highAvailabilityEnabled: false
    storageDir: s3://<s3_bucket>/flink/autoscaling/ha/
    replicas: 1
    resource:
      memory: "1024m"
      cpu: 0.5
  taskManager:
    resource:
      memory: "1024m"
      cpu: 0.5
  job:
    jarURI: s3://<s3_bucket>/some-job-with-back-pressure
    parallelism: 1
    upgradeMode: stateless
```

Para simular la contrapresión, utilice la siguiente especificación de implementación.

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

```
upgradeMode: stateless
```

Cargue el siguiente script de Python en el bucket de S3.

```
import logging
import sys
import time
import random

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

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

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

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

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

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

Para comprobar que la estimación del tiempo de reescalado funciona, asegúrese de que el registro de nivel DEBUG del operador Flink esté habilitado. En el siguiente ejemplo se muestra cómo actualizar el archivo `values.yaml` del gráfico de Helm. A continuación, vuelva a instalar el gráfico de Helm actualizado y vuelva a ejecutar el trabajo de Flink.

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

Obtenga el nombre de tu pod líder.

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

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

Ejecute el siguiente comando para obtener el tiempo de reinicio real utilizado en las evaluaciones de las métricas.

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

Deben aparecer registros similares a los siguientes. Tenga en cuenta que solo el primer escalado utiliza `job.autoscaler.restart.time`. Los escalados posteriores utilizan el tiempo de reinicio observado.

```
2024-05-16 17:17:32,590 o.a.f.a.ScalingExecutor [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT2M
2024-05-16 17:19:03,787 o.a.f.a.ScalingExecutor [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:19:18,976 o.a.f.a.ScalingExecutor [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:20:50,283 o.a.f.a.ScalingExecutor [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
```

```
2024-05-16 17:22:21,691 o.a.f.a.ScalingExecutor [DEBUG][default/autoscaler-example] Restart time used in scaling summary computation: PT14S
```

7.0.0 and 7.1.0

El escalador automático Flink integrado de código abierto utiliza numerosas métricas para tomar las mejores decisiones de escalado. Sin embargo, los valores predeterminados que utiliza para sus cálculos están pensados para ser aplicables a la mayoría de las cargas de trabajo y es posible que no sean óptimos para un trabajo determinado. La característica de ajuste automático añadida a la versión Amazon EMR en EKS del Flink Operator analiza las tendencias históricas observadas en métricas capturadas específicas y, en consecuencia, trata de calcular el valor más óptimo adaptado al trabajo en cuestión.

Configuración	Obligatorio	Predeterminado/a	Descripción
kubernetes.operator.job.autoscaler.autotune.enable	No	False	Indica si el escalador automático de Flink debe ajustar automáticamente las configuraciones a lo largo del tiempo para optimizar las decisiones de escalado de los escaladores automáticos. Actualmente, el escalador automático solo puede ajustar automáticamente el parámetro restart.time del escalador automático.
kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count	No	3	Indica cuántas métricas históricas de Amazon EMR en EKS guarda el escalador automático en el mapa de configuración de métricas de Amazon EMR en EKS.

Configuración	Obligatorio	Predeterminado/a	Descripción
kubernetes.operator.job.autoscaler.autotune.metrics.restart.count	No	3	Indica el número de reinicios que realiza el escalador automático antes de empezar a calcular el tiempo medio de reinicio de un trabajo determinado.

Para habilitar el ajuste automático, debe haber completado lo siguiente:

- Establezca `kubernetes.operator.job.autoscaler.autotune.enable`: en `true`
- Establezca `metrics.job.status.enable`: en `TOTAL_TIME`
- Se sigue la configuración del [Uso del escalador automático para las aplicaciones de Flink](#) para habilitar el escalado automático.

El siguiente es un ejemplo de especificación de implementación que puede utilizar para probar el ajuste automático.

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

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

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

```

jobmanager.scheduler: adaptive

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

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

```

Para simular la contrapresión, utilice la siguiente especificación de implementación.

```

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

```

Cargue el siguiente script de Python en el bucket de S3.

```

import logging
import sys
import time
import random

from pyflink.datastream import StreamExecutionEnvironment

```

```

from pyflink.table import StreamTableEnvironment

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

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

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

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

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

```

Para comprobar que el ajustador automático funciona, utilice los siguientes comandos. Tenga en cuenta que debe utilizar la información de su propio pod líder para el Flink Operator.

Obtenga primero el nombre de su pod líder.

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

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

Una vez que tenga el nombre de su pod líder, puede ejecutar el siguiente comando.

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

Deben aparecer registros similares a los siguientes.

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

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

Mantenimiento y solución de problemas para trabajos de Flink en Amazon EMR en EKS

En las siguientes secciones, se explica cómo realizar un mantenimiento de los trabajos de Flink que llevan mucho tiempo funcionando y se proporcionan instrucciones sobre cómo solucionar algunos problemas comunes con trabajos de Flink.

Mantenimiento de aplicaciones de Flink

Temas

- [Modos de actualización](#)

Las aplicaciones de Flink suelen estar diseñadas para ejecutarse durante largos periodos de tiempo, como semanas, meses o incluso años. Como ocurre con todos los servicios de larga duración,

las aplicaciones de transmisión de Flink deben mantenerse. Esto incluye correcciones de errores, mejoras y migración a un clúster de Flink de una versión posterior.

Cuando cambien las especificaciones para los recursos `FlinkDeployment` y `FlinkSessionJob`, es necesario actualizar la aplicación en ejecución. Para ello, el operador detiene el trabajo en ejecución (a menos que ya esté suspendido) y lo vuelve a implementar con las especificaciones más recientes y, en el caso de las aplicaciones con estado, con el estado de la ejecución anterior.

Los usuarios controlan cómo administrar el estado en el que las aplicaciones con estado se detienen y se restauran con la configuración `upgradeMode` del `JobSpec`.

Modos de actualización

Introducción opcional

Sin estado

Las aplicaciones sin estado se actualizan desde un estado vacío.

Último estado

Las actualizaciones rápidas en cualquier estado de la aplicación (incluso para trabajos con errores) no requieren un trabajo en buen estado, ya que siempre se utiliza el último punto de control correcto. Si se pierden los metadatos de alta disponibilidad, puede ser necesaria una recuperación manual. Para limitar el tiempo que el trabajo puede demorar al seleccionar el último punto de control, puede configurar `kubernetes.operator.job.upgrade.last-state.max.allowed.checkpoint.age`. Si el punto de control es anterior al valor configurado, se utilizará un punto de almacenamiento para los trabajos en buen estado. Esto no se admite en el modo de sesión.

Punto de almacenamiento

Utilice el punto de almacenamiento para la actualización, ya que proporciona la máxima seguridad y la posibilidad de servir como punto de respaldo o bifurcación. El punto de almacenamiento se creará durante el proceso de actualización. Tenga en cuenta que el trabajo de Flink debe estar en ejecución para permitir la creación del punto de almacenamiento. Si el trabajo está en mal estado, se utilizará el último punto de control (a menos que `kubernetes.operator.job.upgrade.last-state-fallback.enabled` tiene el valor `false`). Si el último punto de control no está disponible, la actualización del trabajo fallará.

Solución de problemas

En esta sección, se describe cómo solucionar problemas con Amazon EMR en EKS. Para obtener información sobre cómo solucionar problemas generales con Amazon EMR, consulte [Solucionar problemas de clústeres](#) en la Guía de administración de Amazon EMR.

- [Solución de problemas en trabajos que utilizan PersistentVolumeClaims \(PVC\)](#)
- [Solución de problemas en el escalado automático vertical de Amazon EMR en EKS](#)
- [Solución de problemas del operador de Spark de Amazon EMR en EKS](#)

Solución de problemas de Apache Flink en Amazon EMR en EKS

No se encontró la asignación de recursos al instalar el gráfico de Helm

Es posible que aparezca el siguiente mensaje de error cuando instale el gráfico de Helm.

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

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

ensure CRDs are installed first].
```

Para resolver este error, instale cert-manager para habilitar la adición del componente webhook. Debe instalar cert-manager a cada clúster de Amazon EKS que utilice.

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

Servicio de AWS error de acceso denegado

Si ve un acceso denegado error, confirme que la función de IAM del `values.yaml` archivo `operatorExecutionRoleArn` de gráficos de Helm tiene los permisos correctos. Asegúrese

también de que el rol de IAM asignado a `executionRoleArn` en su especificación `FlinkDeployment` tenga los permisos correctos.

FlinkDeployment está atascada

Si se bloquea la `FlinkDeployment`, siga estos pasos para forzar la eliminación de la implementación:

1. Edite la ejecución de la implementación.

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

2. Elimine este finalizador.

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

3. Elimine la implementación.

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

El `AWSBad RequestException` problema s3a se produce al ejecutar una aplicación Flink de forma opcional Región de AWS

Si ejecutas una aplicación de Flink de forma [opcional Región de AWS](#), es posible que veas los siguientes errores:

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

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

```
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
AHl42uDNaTUF0us/5IIVNvSakBcMjMCH7dd37ky0vE6jhABCDEFGHIJKLMN0PQRST==)
```

Para corregir estos errores, utilice la siguiente configuración en el archivo de definición `FlinkDeployment`.

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

También le recomendamos que utilice el proveedor de SDKv2 credenciales:

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

Si quieres usar el proveedor de SDKv1 credenciales, asegúrate de que tu SDK sea compatible con tu región de suscripción. Para obtener más información, consulta el [aws-sdk-java GitHub repositorio](#).

Si obtiene `S3 AWSBadRequestException` cuando ejecuta instrucciones SQL de Flink en una región opcional, asegúrese de establecer la configuración `fs.s3a.endpoint.region`: *OPT_IN_AWS_REGION_NAME* en sus especificaciones de configuración de Flink.

`S3A AWSBad RequestException` cuando se ejecuta un trabajo de sesión de Flink en regiones CN

En el caso de las versiones 6.15.0 a 7.2.0 de Amazon EMR, es posible que aparezca los siguientes mensajes de error al ejecutar un trabajo de sesión de Flink en las regiones de CN. Estas incluyen China (Pekín) y China (Ningxia):

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

[{"type":"org.apache.hadoop.fs.s3a.AWSBadRequestException","message":"getFileStatus on
s3://ABCDPath: software.amazon.awssdk.services.s3.model.S3Exception:
      null (Service: S3, Status Code: 400, Request ID: ABCDEFGH, Extended
Request ID: ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID: ABCDEFGH,
```

```

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

```

Existe conciencia sobre este problema. El equipo está trabajando para parchear los operadores de Flink en todas estas versiones de lanzamiento. Sin embargo, antes de finalizar el parche, para corregir este error, necesita descargar el gráfico de Helm del operador de Flink, descomprimirlo (extraer el archivo comprimido) y realizar cambios de configuración en el gráfico de Helm.

A continuación se indican los pasos a seguir:

1. Cambie a su carpeta local para el gráfico de Helm, específicamente cambie los directorios a, y ejecute la siguiente línea de comandos para extraer el gráfico de Helm y descomprimirlo (extraerlo).

```

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

```

```

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

```

2. Vaya a la carpeta del gráfico de Helm y busque el archivo `templates/flink-operator.yaml`.
3. Busque `flink-operator-config ConfigMap` y añada la siguiente `fs.s3a.endpoint.region` configuración en `flink-conf.yaml` Por ejemplo:

```

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

```

4. Instale el gráfico de Helm local y ejecute su trabajo.

Versiones de Amazon EMR en EKS compatibles con Apache Flink

Apache Flink está disponible con las siguientes versiones de Amazon EMR en EKS. Para obtener información sobre todas las versiones disponibles, consulte [Versiones de Amazon EMR en EKS](#).

Etiqueta de la versión	Java	Flink	Operador de Flink
emr-7.2.0-flink-latest	17	1,1181	-
emr-7.2.0-flink-k8 s-operator-latest	11	-	1.8.0
emr-7.1.0-flink-latest	17	1.18.1	-
emr-7.1.0-flink-k8 s-operator-latest	11	-	1.6.1
emr-7.0.0-flink-latest	11	1.18.0	-
emr-7.0.0-flink-k8 s-operator-latest	11	-	1.6.1
emr-6.15.0-flink-latest	11	1.17.1	-
emr-6.15.0-flink-k8 s-operator-latest	11	-	1.6.0
emr-6.14.0-flink-latest	11	1.17.1	-
emr-6.14.0-flink-k8 s-operator-latest	11	-	1.6.0
emr-6.13.0-flink-latest	11	1.17.0	-
emr-6.13.0-flink-k8 s-operator-latest	11	-	1.5.0

Ejecución de trabajos de Spark con Amazon EMR en EKS

Una ejecución de trabajo es una unidad de trabajo, como un jar de Spark, un PySpark script o una consulta de SparkSQL, que se envía a Amazon EMR en EKS. En este tema se proporciona información general sobre la administración de ejecuciones de tareas mediante la consola Amazon EMR AWS CLI, la visualización de las ejecuciones de tareas mediante la consola Amazon EMR y la solución de errores comunes de ejecución de tareas.

Ten en cuenta que no puedes ejecutar trabajos de IPv6 Spark en Amazon EMR en EKS

Note

Antes de enviar una ejecución de trabajo con Amazon EMR en EKS, debe completar los pasos que se indican en [Configuración de Amazon EMR en EKS](#).

Temas

- [Ejecución de trabajos de Spark con StartJobRun](#)
- [Ejecución de trabajos de Spark con el operador de Spark](#)
- [Ejecución de trabajos de Spark con spark-submit](#)
- [Uso de Apache Livy con Amazon EMR en EKS](#)
- [Administración de las ejecuciones de trabajos de Amazon EMR en EKS](#)
- [Uso de plantillas de trabajos](#)
- [Uso de plantillas de pods](#)
- [Uso de políticas de reintento de trabajos](#)
- [Uso de la rotación del registro de eventos de Spark](#)
- [Uso de la rotación de los registros de contenedores de Spark](#)
- [Uso del escalado automático vertical con trabajos de Spark de Amazon EMR](#)

Ejecución de trabajos de Spark con **StartJobRun**

Esta sección incluye pasos de configuración detallados para preparar tu entorno para ejecutar trabajos de Spark y, a continuación, proporciona step-by-step instrucciones para enviar una ejecución de trabajo con parámetros específicos.

Temas

- [Configuración de Amazon EMR en EKS](#)
- [Enviar una ejecución de trabajo con StartJobRun](#)
- [Uso de la clasificación de remitentes de trabajos](#)
- [Uso de la clasificación por defecto del contenedor Amazon EMR](#)

Configuración de Amazon EMR en EKS

Complete las siguientes tareas para la configuración de Amazon EMR en EKS. Si ya se ha registrado en Amazon Web Services (AWS) y ha estado usando Amazon EKS, ya lo tiene todo casi listo para usar Amazon EMR en EKS. Omita cualquier tarea que ya haya completado.

Note

También puede seguir el [Taller de Amazon EMR en EKS](#) para configurar todos los recursos necesarios y ejecutar trabajos de Spark en Amazon EMR en EKS. El taller también proporciona automatización mediante el uso de CloudFormation plantillas para crear los recursos necesarios para empezar. Para ver otras plantillas y prácticas recomendadas, consulte nuestra [Guía de mejores prácticas para contenedores EMR](#) en GitHub.

1. [Instale o actualice a la última versión de AWS CLI](#)
2. [Configuración de kubectl y eksctl](#)
3. [Introducción a Amazon EKS: eksctl](#)
4. [Habilite el acceso al clúster para Amazon EMR en EKS](#)
5. [Habilite las funciones de IAM para el clúster de EKS](#)
6. [Otorgar a los usuarios acceso a Amazon EMR en EKS](#)
7. [Registrar el clúster de Amazon EKS con Amazon EMR](#)

Habilitar el acceso a clústeres de Amazon EMR en EKS

Las siguientes secciones muestran un par de formas de habilitar el acceso a los clústeres. La primera es mediante la administración de acceso a clústeres (CAM) de Amazon EKS y la segunda muestra cómo tomar medidas manuales para habilitar el acceso a los clústeres.

Habilitación del acceso al clúster mediante EKS Access Entry (recomendado)

Note

aws-auth ConfigMap se ha quedado obsoleto. [El método recomendado para administrar el acceso a Kubernetes APIs son las entradas de acceso.](#)

Amazon EMR está integrado con la [administración de acceso a clústeres \(CAM\) de Amazon EKS](#), por lo que puede automatizar la configuración de las políticas AuthN y AuthZ necesarias para ejecutar trabajos de Amazon EMR Spark en los espacios de nombres de los clústeres de Amazon EKS. Cuando crea un clúster virtual a partir de un espacio de nombres de clústeres de Amazon EKS, Amazon EMR configura automáticamente todos los permisos necesarios, por lo que no necesita añadir ningún paso adicional a sus flujos de trabajo actuales.

Note

La integración de Amazon EMR con Amazon EKS CAM solo es compatible con los nuevos clústeres virtuales de Amazon EMR en EKS. No puede migrar los clústeres virtuales existentes para utilizar esta integración.

Requisitos previos

- Asegúrese de que está ejecutando la versión 2.15.3 o superior del AWS CLI
- El clúster de Amazon EKS debe estar en la versión 1.23 o posteriores.

Configuración

Para configurar la integración entre Amazon EMR y las operaciones de AccessEntry API de Amazon EKS, asegúrese de haber completado los siguientes pasos:

- Asegúrese de que authenticationMode de su clúster de Amazon EKS esté establecido en API_AND_CONFIG_MAP.

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

Si aún no lo está, establezca authenticationMode en API_AND_CONFIG_MAP.

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

Para obtener más información acerca de los modos de autenticación, consulte [Modos de autenticación de clústeres](#).

- Asegúrese de que el [rol de IAM](#) que está utilizando para ejecutar las operaciones de la API `CreateVirtualCluster` y `DeleteVirtualCluster` también tenga los siguientes permisos:

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

Conceptos y terminología

La siguiente es una lista de terminologías y conceptos relacionados con Amazon EKS CAM.

- **Clúster virtual (VC):** representación lógica del espacio de nombres creado en Amazon EKS. Es un enlace 1:1 a un espacio de nombres de un clúster de Amazon EKS. Puede usarlo para ejecutar cargas de trabajo de Amazon EMR en un clúster de Amazon EKS dentro del espacio de nombres especificado.

- Espacio de nombres: mecanismo para aislar grupos de recursos dentro de un único clúster de EKS.
- Política de acceso: permisos que otorgan acceso y acciones a un rol de IAM dentro de un clúster de EKS.
- Entrada de acceso: una entrada creada con un ARN de rol. Puede vincular la entrada de acceso a una política de acceso para asignar permisos específicos en el clúster de Amazon EKS.
- Clúster virtual integrado de entrada de acceso EKS: el clúster virtual creado mediante las [operaciones de la API de entrada de acceso](#) desde Amazon EKS.

Habilitación del acceso al clúster mediante **aws-auth**

Debe permitir que Amazon EMR en EKS acceda a un espacio de nombres específico de su clúster mediante las siguientes acciones: crear un rol de Kubernetes, vincular el rol a un usuario de Kubernetes y asignar el usuario de Kubernetes con el rol [AWSServiceRoleForAmazonEMRContainers](#) vinculado al servicio. Estas acciones se automatizan en `eksctl` cuando se utiliza el comando de asignación de identidades de IAM con `emr-containers` como nombre del servicio. Puede llevar a cabo estas operaciones fácilmente con el siguiente comando.

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

my_eks_cluster Sustitúyalo por el nombre del clúster de Amazon EKS y *kubernetes_namespace* sustitúyalo por el espacio de nombres de Kubernetes creado para ejecutar cargas de trabajo de Amazon EMR.

Important

Para descargar la versión más reciente de `eksctl`, siga el paso anterior [Configuración de kubectl y eksctl](#) para utilizar esta funcionalidad.

Pasos manuales para habilitar el acceso a clústeres de Amazon EMR en EKS

También puede utilizar los siguientes pasos manuales para habilitar el acceso al clúster de Amazon EMR en EKS.

1. Crear un rol de Kubernetes en un espacio de nombres específico

Amazon EKS 1.22 - 1.29

Con las versiones de Amazon EKS de 1.22 a 1.29, ejecute el siguiente comando para crear un rol de Kubernetes en un espacio de nombres específico. Este rol concede los permisos de RBAC necesarios a Amazon EMR en EKS.

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

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

Amazon EKS 1.21 and below

Con Amazon EKS 1.21 y versiones anteriores, ejecute el siguiente comando para crear un rol de Kubernetes en un espacio de nombres específico. Este rol concede los permisos de RBAC necesarios a Amazon EMR en EKS.

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

```

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

```

2. Crear una vinculación de roles de Kubernetes limitada al espacio de nombres

Ejecute el siguiente comando para crear una vinculación de roles de Kubernetes en el espacio de nombres dado. Esta vinculación de roles otorga los permisos definidos en el rol creado en el paso anterior a un usuario nombrado `emr-containers`. Este usuario identifica los [roles vinculadas al servicio de Amazon EMR en EKS](#) y, por lo tanto, permite que Amazon EMR en EKS lleve a cabo las acciones definidas por el rol que creó.

```

namespace=my-namespace

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

```

3. Actualizar el mapa de configuración **aws-auth** de Kubernetes

Puede utilizar una de las siguientes opciones para asignar el rol vinculado al servicio de Amazon EMR en EKS con el usuario de `emr-containers` que estaba vinculado con el rol de Kubernetes en el paso anterior.

Opción 1: con **eksctl**

Ejecute el siguiente comando `eksctl` para asignar el rol vinculado al servicio de Amazon EMR en EKS con el usuario `emr-containers`.

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

Opción 2: sin usar `eksctl`

1. Ejecute el siguiente comando para abrir el mapa de configuración `aws-auth` en el editor de texto.

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

Note

Si recibe un mensaje de error al respecto `Error from server (NotFound): configmaps "aws-auth" not found`, consulte los pasos de [Añadir funciones de usuario](#) en la Guía del usuario de Amazon EKS para aplicar el stock ConfigMap.

2. Agregue los detalles del rol vinculado al servicio Amazon EMR en EKS en la sección `mapRoles` de ConfigMap, en `data`. Agregue esta sección si no existe todavía en el archivo. La sección `mapRoles` actualizada en datos es similar al siguiente ejemplo.

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

3. Guarde el archivo y salga del editor de texto.

Habilite las funciones de IAM para el clúster de EKS

En los siguientes temas se detallan las opciones para habilitar las funciones de IAM.

Temas

- [Opción 1: Habilitar la identidad del pod de EKS en el clúster de EKS](#)
- [Opción 2: habilitar las funciones de IAM para las cuentas de servicio \(IRSA\) en el clúster EKS](#)

Opción 1: Habilitar la identidad del pod de EKS en el clúster de EKS

Las asociaciones de identidad de los pods de Amazon EKS permiten gestionar las credenciales de sus aplicaciones, de forma similar a como los perfiles de EC2 instancias de Amazon proporcionan credenciales a las EC2 instancias de Amazon. Pod Identity de Amazon EKS proporciona credenciales a sus cargas de trabajo con una API de autenticación de EKS adicional y un pod de agente que se ejecuta en cada nodo.

Amazon EMR en EKS comienza a admitir la identidad de pod de EKS desde la versión emr-7.3.0 para el modelo de envío. StartJobRun

Para obtener más información sobre las identidades de los pods de EKS, consulte [Cómo funciona la identidad de los pods de EKS](#).

¿Por qué elegir EKS Pod Identities?

Como parte de la configuración de EMR, la función de ejecución de tareas debe establecer límites de confianza entre una función de IAM y las cuentas de servicio en un espacio de nombres específico (de clústeres virtuales de EMR). Con IRSA, esto se logró mediante la actualización de la política de confianza de la función de ejecución de tareas de EMR. Sin embargo, debido al límite estricto de 4096 caracteres en la longitud de la política de confianza de IAM, existía la restricción de compartir un único rol de IAM de Job Execution en un máximo de doce (12) clústeres de EKS.

Gracias a la compatibilidad de EMR con las identidades de pod, el equipo de EKS gestiona ahora el límite de confianza entre las funciones de IAM y las cuentas de servicio mediante la asociación entre las funciones de IAM y las cuentas de servicio. APIs

 Note

El límite de seguridad de la identidad de los pods de EKS sigue estando en el nivel de la cuenta de servicio, no en el nivel del pod.

Consideraciones sobre la identidad del pod

Para obtener información sobre las limitaciones de identidad del pod, consulte [Consideraciones sobre la identidad del pod de EKS](#).

Prepare la identidad del pod de EKS en el clúster de EKS

Compruebe si el permiso requerido existe en `NodeInstanceRole`

El rol de nodo `NodeInstanceRole` necesita un permiso para que el agente realice la `AssumeRoleForPodIdentity` acción en la API de autenticación de EKS. Puede añadir lo siguiente a [Amazon EKSWorker NodePolicy](#), tal como se define en la Guía del usuario de Amazon EKS, o utilizar una política personalizada.

Si su clúster de EKS se creó con una versión `eksctl` superior a la 0.181.0, Amazon `EKSWorkerNodePolicy`, incluido el `AssumeRoleForPodIdentity` permiso requerido, se asociará automáticamente al rol de nodo. Si el permiso no está presente, añada manualmente el siguiente permiso a `Amazon EKSWorker NodePolicy` que te permita asumir un rol para la identidad del pod. El agente de identidad del pod de EKS necesita este permiso para recuperar las credenciales de los pods.

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

Cree el complemento EKS pod Identity Agent

Usa el siguiente comando para crear el complemento EKS Pod Identity Agent con la última versión:

```
aws eks create-addon --cluster-name cluster-name --addon-name eks-pod-identity-agent
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Siga los siguientes pasos para crear el complemento EKS Pod Identity Agent desde la consola Amazon EKS:

1. Abra la consola Amazon EKS: consola [Amazon EKS](#).
2. En el panel de navegación izquierdo, seleccione Clústeres y, a continuación, seleccione el nombre del clúster para el que desea configurar el complemento de agente de Pod Identity de EKS.
3. Elija la pestaña Complementos.
4. Escoja Obtener más complementos.
5. Seleccione la casilla situada en la parte superior derecha del cuadro de complementos para el agente de Pod Identity de EKS y, a continuación, elija Siguiente.
6. En la página Configurar los ajustes de los complementos seleccionados, seleccione cualquier versión en la lista desplegable de versiones.
7. (Opcional) Expanda Valores de configuración opcionales para introducir una configuración adicional. Por ejemplo, puede proporcionar una ubicación de imagen de contenedor alternativa e ImagePullSecrets. El esquema JSON con las claves aceptadas se muestra en Esquema de configuración de complementos.

Introduzca las claves y los valores de configuración en Valores de configuración.

8. Elija Siguiente.
9. Confirme que los pods de agente se estén ejecutando en su clúster mediante la CLI.

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

Un ejemplo de resultado es el siguiente:

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

Esto configura un nuevo DaemonSet en el kube-system espacio de nombres. El agente de identidad del pod de Amazon EKS, que se ejecuta en cada nodo de EKS, utiliza la [AssumeRoleForPodIdentity](#) acción para recuperar las credenciales temporales de la API de autenticación de EKS. Luego, estas credenciales están disponibles para las AWS SDKs que ejecute dentro de sus contenedores.

Para obtener más información, consulte el requisito previo del documento público: [Configurar el Amazon EKS Pod Identity](#) Agent.

Crear un rol de ejecución de tareas

Cree o actualice el rol de ejecución de trabajos que permita EKS Pod Identity

Para ejecutar cargas de trabajo con Amazon EMR en EKS, debe crear un rol de IAM. En esta documentación, nos referimos a este rol como rol de ejecución de trabajos. Para obtener más información sobre cómo crear el rol de IAM, consulte [Creación de roles de IAM](#) en la guía del usuario.

Además, debe crear una política de IAM que especifique los permisos necesarios para la función de ejecución de tareas y, a continuación, adjuntar esta política a la función para habilitar EKS Pod Identity.

Por ejemplo, tiene la siguiente función de ejecución de tareas. Para obtener más información, consulte [Crear un rol de ejecución de tareas](#).

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

Important

Amazon EMR en EKS crea automáticamente cuentas de servicio de Kubernetes en función del nombre de su función de ejecución de tareas. Asegúrese de que el nombre del rol no sea demasiado largo, ya que su trabajo podría fallar si la combinación de `cluster_namepod_name`, y `service_account_name` supera el límite de longitud.

Configuración del rol de ejecución del trabajo: asegúrese de que el rol de ejecución del trabajo se cree con el siguiente permiso de confianza para EKS Pod Identity. Para actualizar un rol de ejecución de tareas existente, configúrelo para que confíe en el siguiente director de servicio de EKS como permiso adicional de la política de confianza. Este permiso de confianza puede coexistir con las políticas de confianza de IRSA existentes.

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

Permiso de usuario: los usuarios necesitan el `iam:PassRole` permiso para ejecutar llamadas a la `StartJobRun` API o enviar trabajos. Este permiso permite a los usuarios transferir la función de ejecución de tareas a EMR en EKS. Los administradores de tareas deberían tener el permiso de forma predeterminada.

A continuación se muestra el permiso que necesita un usuario:

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

Para restringir aún más el acceso del usuario a clústeres de EKS específicos, añada el filtro de `AssociatedResourceArn` atributos a la política de IAM. Limita la asunción de funciones a los clústeres de EKS autorizados, lo que refuerza los controles de seguridad a nivel de recursos.

```
"Condition": {
```

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

Configure las asociaciones de identidad de los pods de EKS

Requisito previo

Asegúrese de que la identidad de IAM que crea la asociación de identidades del pod, como un usuario administrador de EKS, tenga el permiso `eks:CreatePodIdentityAssociation` y `iam:PassRole`.

```
{
  "Effect": "Allow",
  "Action": [
    "eks:CreatePodIdentityAssociation",
  ],
  "Resource": "* or role-arn"
},
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "* or role-arn",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "pods.eks.amazonaws.com"
        }
      }
    }
  ]
}
```

Crear asociaciones para el rol y la cuenta de servicio EMR

Create EMR role associations through the AWS CLI

Al enviar un trabajo a un espacio de nombres de Kubernetes, un administrador debe crear asociaciones entre la función de ejecución del trabajo y la identidad de la cuenta de servicio gestionado de EMR. Tenga en cuenta que la cuenta de servicio administrado de EMR se crea

automáticamente en el momento del envío del trabajo y tiene como límite el espacio de nombres donde se envía el trabajo.

Con la versión 2.24.0 AWS CLI (anterior), ejecuta el siguiente comando para crear asociaciones de roles con la identidad del pod.

Ejecuta el siguiente comando para crear asociaciones de roles con la identidad del pod:

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

Nota:

- Cada clúster puede tener un límite de 1000 asociaciones. Cada función de ejecución de tareas (mapeo de espacios de nombres) requerirá 3 asociaciones para los módulos de remitente, controlador y ejecutor del trabajo.
- Solo puedes asociar roles que estén en la misma AWS cuenta que el clúster. Puede delegar el acceso desde otra cuenta al rol de esta cuenta que haya configurado para que lo utilice Pod Identities de EKS. Para ver un tutorial sobre cómo delegar el acceso `AssumeRole`, consulte el [tutorial de IAM: Delegar el acceso entre AWS cuentas mediante funciones de IAM](#).

Create EMR role associations through Amazon EKS

EMR crea una cuenta de servicio con un patrón de nombres determinado cuando se envía un trabajo. Para realizar asociaciones manuales o integrar este flujo de trabajo con el AWS SDK, siga estos pasos:

Cree el nombre de la cuenta de servicio:

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

Los siguientes ejemplos crean una asociación de roles para un ejemplo de rol de ejecución de tareas `JobExecutionRoleIRSAv2`.

Ejemplos de asociaciones de roles:

```
RoleName: JobExecutionRoleIRSAv2
```

```
Base36EncodingOfRoleName: 2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

Ejemplo de comando CLI:

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

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

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

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

Una vez que haya completado todos los pasos necesarios para la identidad del pod de EKS, puede omitir los siguientes pasos para configurar el IRSA:

- [Habilite las funciones de IAM para las cuentas de servicio \(IRSA\) en el clúster de EKS](#)
- [Cree un rol de ejecución de tareas](#)
- [Actualice la política de confianza del rol de ejecución del trabajo](#)

Puede pasar directamente al siguiente paso: [Conceder a los usuarios acceso a Amazon EMR en EKS](#)

Eliminar asociaciones de roles

Siempre que elimine un clúster virtual o un rol de ejecución de tareas y ya no desee dar acceso a EMR a sus cuentas de servicio, debe eliminar las asociaciones del rol. Esto se debe a que EKS

permite las asociaciones con recursos inexistentes (espacio de nombres y cuenta de servicio). Amazon EMR en EKS recomienda eliminar las asociaciones si se elimina el espacio de nombres o si el rol ya no se usa, a fin de liberar espacio para otras asociaciones.

 Note

Si no las elimina, las asociaciones persistentes podrían afectar a su capacidad de escalado, ya que EKS tiene limitaciones en cuanto al número de asociaciones que puede crear (límite máximo: 1000 asociaciones por clúster). Puedes enumerar las asociaciones de identidad de los pods en un espacio de nombres determinado para comprobar si hay alguna asociación persistente que deba limpiarse:

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

Con la AWS CLI versión 2.24.0 o superior, ejecuta el siguiente comando `emr-containers` para eliminar las asociaciones de funciones de EMR:

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

Migre automáticamente el IRSA existente a Pod Identity

Puede utilizar la herramienta `eksctl` para migrar las funciones de IAM existentes para las cuentas de servicio (IRSA) a las asociaciones de identidad de los pods:

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

Al ejecutar el comando sin la `--approve` marca, solo se generará un plan que refleje los pasos de la migración y no se producirá ninguna migración real.

Solución de problemas

Mi trabajo ha fallado con el proveedor de credenciales `NoClassDefinitionFound` o con `ClassNotFound` una excepción, o no he podido obtener el proveedor de credenciales.

EKS Pod Identity utiliza el proveedor de credenciales del contenedor para recuperar las credenciales necesarias. Si ha especificado un proveedor de credenciales personalizado, asegúrese de que funciona correctamente. Como alternativa, asegúrate de usar una versión correcta AWS del SDK que sea compatible con EKS Pod Identity. Para obtener más información, consulte [Comenzar con Amazon EKS](#).

El trabajo falló y el error «No se pudieron recuperar las credenciales debido al límite de tamaño [x]» que aparece en el `eks-pod-identity-agent` registro.

EMR en EKS crea cuentas de servicio de Kubernetes en función del nombre del rol de ejecución del trabajo. Si el nombre del rol es demasiado largo, EKS Auth no podrá recuperar las credenciales porque la combinación de `cluster_namepod_name`, y `service_account_name` supera el límite de longitud. Identifique qué componente ocupa más espacio y ajuste el tamaño en consecuencia.

El trabajo falló y en el `eks-pod-identity` registro aparece el error «No se pudieron recuperar las credenciales xxx».

Una posible causa de este problema podría ser que el clúster EKS esté configurado en subredes privadas sin PrivateLink configurarlo correctamente. Compruebe si el clúster está en una red privada y AWS PrivateLink configúrelo para solucionar el problema. Para obtener instrucciones detalladas, consulte [Comenzar a usar Amazon EKS](#).

Opción 2: habilitar las funciones de IAM para las cuentas de servicio (IRSA) en el clúster EKS

La característica de roles de IAM para cuentas de servicio está disponible en los nuevos clústeres de la versión 1.14 de Amazon EKS y posteriores, y para clústeres de EKS que se actualizaron a las versiones 1.13 o posteriores a partir del 3 de septiembre de 2019. Para utilizar esta característica, puede actualizar los clústeres de EKS existentes a la versión 1.14 o posteriores. Para obtener más información, consulte [Actualización de una versión de Kubernetes de clúster de Amazon EKS](#).

Si el clúster admite roles de IAM para cuentas de servicio, tiene asociada una URL de emisor de [OpenID Connect](#). Puede ver esta URL en la consola de Amazon EKS o utilizar el siguiente AWS CLI comando para recuperarla.

⚠ Important

Debe usar la última versión de AWS CLI para recibir el resultado correcto de este comando.

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

El resultado esperado es el siguiente.

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

Para utilizar roles de IAM para cuentas de servicio en su clúster, debe crear un proveedor de identidad OIDC con [eksctl](#) o la [AWS Management Console](#).

A fin de crear un proveedor de identidad de OIDC de IAM para su clúster con **eksctl**

Consulte su versión de **eksctl** con el siguiente comando. En este procedimiento, se presupone que ha instalado **eksctl** y que su versión de **eksctl** es al menos 0.32.0 o posterior.

```
eksctl version
```

Para obtener más información sobre cómo instalar o actualizar **eksctl**, consulte [Instalación o actualización de eksctl](#).

Cree su proveedor de identidad OIDC para su clúster con el siguiente comando. Reemplace *cluster_name* por su propio valor.

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

Para crear un proveedor de identidades OIDC de IAM para su clúster con AWS Management Console

Recupere la URL del emisor del OIDC de la descripción de su clúster en la consola Amazon EKS o utilice el siguiente comando. AWS CLI

Utilice el siguiente comando para recuperar la URL del emisor de OIDC de la AWS CLI.

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

Siga estos pasos para recuperar la URL del emisor de OIDC de la consola de Amazon EKS.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Proveedores de identidad y, a continuación, elija Crear un proveedor.
 1. En Provider Type (Tipo de proveedor), elija Choose a provider type (Elegir un tipo de proveedor) y, luego, elija OpenID Connect.
 2. En Provider URL (URL del proveedor), pegue la URL del emisor OIDC de su clúster.
 3. En Audiencia, ingrese sts.amazonaws.com y seleccione Paso siguiente.
3. Compruebe que la información del proveedor es correcta y, a continuación, seleccione Create (Crear) para crear su proveedor de identidad.

Crear un rol de ejecución de trabajos

Para ejecutar cargas de trabajo en Amazon EMR en EKS, debe crear un rol de IAM. En esta documentación, nos referimos a este rol como rol de ejecución de trabajos. 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.

También debe crear una política de IAM que especifique los permisos para el rol de ejecución de trabajos y, a continuación, adjuntar la política de IAM a dicho rol.

La siguiente política para la función de ejecución de tareas permite el acceso a los objetivos de recursos, Amazon S3 y CloudWatch. Estos permisos son necesarios para supervisar los trabajos y acceder a los registros. Para seguir el mismo proceso con el AWS CLI, también puede configurar su función siguiendo los pasos de la sección [Crear una función de IAM para la ejecución de tareas](#) del taller Amazon EMR on EKS.

Note

El acceso debe tener el alcance adecuado y no debe concederse a todos los objetos de S3 que desempeñen el rol de ejecución de trabajos.

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

```

    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "s3:PutObject",
          "s3:GetObject",
          "s3:ListBucket"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
      },
      {
        "Effect": "Allow",
        "Action": [
          "logs:PutLogEvents",
          "logs:CreateLogStream",
          "logs:DescribeLogGroups",
          "logs:DescribeLogStreams"
        ],
        "Resource": [
          "arn:aws:logs:*:*:*"
        ]
      }
    ]
  }
}

```

Para obtener más información, consulte [Usar funciones de ejecución de tareas](#), [Configurar una ejecución de tareas para usar registros de S3](#) y [Configurar una ejecución de tareas para usar CloudWatch](#) registros.

Actualizar la política de confianza del rol de ejecución de trabajos

Cuando utiliza los roles de IAM para cuentas de servicio (IRSA) para ejecutar tareas en un espacio de nombres de Kubernetes, el administrador debe crear una relación de confianza entre el rol de ejecución de trabajos y la identidad de la cuenta de servicio administrada de EMR. Para crear la relación de confianza, se puede actualizar la política de confianza del rol de ejecución de trabajos. Tenga en cuenta que la cuenta de servicio administrado de EMR se crea automáticamente en el momento del envío del trabajo y tiene como límite el espacio de nombres donde se envía el trabajo.

Para actualizar la política de confianza, ejecute el siguiente comando.

```
aws emr-containers update-role-trust-policy \
```

```
--cluster-name cluster \  
--namespace namespace \  
--role-name iam_role_name_for_job_execution
```

Para obtener más información, consulte [Uso de roles de ejecución de trabajos con Amazon EMR en EKS](#).

Important

El operador que ejecute el comando anterior debe tener los siguientes permisos:
`eks:DescribeCluster, iam:GetRole, iam:UpdateAssumeRolePolicy`.

Otorgar a los usuarios acceso a Amazon EMR en EKS

Para cualquier acción que lleve a cabo en Amazon EMR en EKS, necesita el permiso de IAM correspondiente. Debe crear una política de IAM que le permita llevar a cabo acciones de Amazon EMR en EKS y adjuntar la política al usuario o rol de IAM que utilice.

En este tema se proporcionan los pasos para crear una política nueva y adjuntarla a un usuario. También cubre los permisos básicos que necesita para configurar su entorno de Amazon EMR en EKS. Le recomendamos que perfeccione los permisos para recursos específicos siempre que sea posible en función de las necesidades de su empresa.

Crear una nueva política de IAM y adjuntarla a un usuario en la consola de IAM

Crear una política de IAM nueva

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación izquierdo de la consola de IAM elija Políticas.
3. En la página Políticas, seleccione Crear una política.
4. En la ventana Crear política, vaya a la pestaña Editar JSON. Cree un documento de política con una o más instrucciones JSON, tal como se muestra en los ejemplos que siguen a este procedimiento. A continuación, seleccione Revisar política.
5. En la pantalla Review Policy (Revisar política), escriba su Policy Name (Nombre de política); por ejemplo, `AmazonEMR0nEKSP0l1cy`. Ingrese una descripción opcional y, a continuación, elija Crear política.

Adjuntar la política a un usuario o rol

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en <https://console.aws.amazon.com/iam/>
2. En el panel de navegación, seleccione Políticas.
3. En la lista de políticas, seleccione la casilla situada junto a la política creada en la sección anterior. Puede utilizar el menú Filtro y el cuadro de búsqueda para filtrar la lista de políticas.
4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Elija el usuario o rol al que adjuntar la política. Puede utilizar el menú Filtro y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario o rol al que adjuntará la política, seleccione Adjuntar política.

Permisos para administrar clústeres virtuales

Para gestionar los clústeres virtuales de su AWS cuenta, cree una política de IAM con los siguientes permisos. Estos permisos le permiten crear, enumerar, describir y eliminar clústeres virtuales en su AWS cuenta.

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

```

    ],
    "Resource": "*"
  }
]
}

```

Amazon EMR está integrado con la administración de acceso a clústeres (CAM) de Amazon EKS, por lo que puede automatizar la configuración de las políticas AuthN y AuthZ necesarias para ejecutar trabajos de Amazon EMR Spark en los espacios de nombres de los clústeres de Amazon EKS. Para llevar a cabo esto, debe tener los siguientes permisos:

```

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

```

Para obtener más información, consulte [Automatización de la habilitación del acceso a clústeres de Amazon EMR en EKS](#).

Cuando se invoque la `CreateVirtualCluster` operación por primera vez desde una AWS cuenta, también necesitará `CreateServiceLinkedRole` los permisos para crear el rol vinculado al servicio para Amazon EMR en EKS. Para obtener más información, consulte [Uso de roles vinculados a servicios para Amazon EMR en EKS](#).

Permisos para enviar trabajos

Para enviar trabajos en los clústeres virtuales de su AWS cuenta, cree una política de IAM con los siguientes permisos. Estos permisos le permiten iniciar, enumerar, describir y cancelar la ejecución de trabajos para todos los clústeres virtuales de su cuenta. Debería considerar la posibilidad de agregar permisos para enumerar o describir los clústeres virtuales, lo que le permitirá comprobar el estado del clúster virtual antes de enviar los trabajos.

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

Permisos de depuración y supervisión

Para acceder a los registros enviados a Amazon S3 o ver los CloudWatch registros de eventos de la aplicación en la consola de Amazon EMR, cree una política de IAM con los siguientes permisos. Le recomendamos que perfeccione los permisos para recursos específicos siempre que sea posible en función de las necesidades de su empresa.

Important

Si no ha creado ningún bucket de Amazon S3, debe agregar el permiso `s3:CreateBucket` a la instrucción de política. Si no ha creado ningún grupo de registros, debe agregar `logs:CreateLogGroup` a la instrucción de política.

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



```

    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "emr-containers:DescribeJobRun",
          "elasticmapreduce:CreatePersistentAppUI",
          "elasticmapreduce:DescribePersistentAppUI",
          "elasticmapreduce:GetPersistentAppUIPresignedURL"
        ],
        "Resource": "*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "s3:GetObject",
          "s3:ListBucket"
        ],
        "Resource": "*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "logs:Get*",
          "logs:DescribeLogGroups",
          "logs:DescribeLogStreams"
        ],
        "Resource": "*"
      }
    ]
  }
}

```

Para obtener más información sobre cómo configurar una ejecución de trabajo para enviar registros a Amazon S3 CloudWatch, consulte [Configurar una ejecución de trabajo para usar registros de S3](#) y [Configurar una ejecución de trabajo para usar CloudWatch registros](#).

Registrar el clúster de Amazon EKS con Amazon EMR

El registro de su clúster es el último paso necesario para configurar Amazon EMR en EKS y ejecutar cargas de trabajo.

Utilice el siguiente comando para crear un clúster virtual con el nombre que desee para el clúster y el espacio de nombres de Amazon EKS que configuró en los pasos anteriores.

Note

Cada clúster virtual debe tener un nombre único en todos los clústeres de EKS. Si dos clústeres virtuales tienen el mismo nombre, el proceso de implementación fallará aunque los dos clústeres virtuales pertenezcan a clústeres de EKS diferentes.

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

Como alternativa, puede crear un archivo JSON que incluya los parámetros necesarios para el clúster virtual y, a continuación, ejecutar el comando `create-virtual-cluster` con la ruta al archivo JSON. Para obtener más información, consulte [Administración de clústeres virtuales](#).

Note

Para validar la creación correcta de un clúster virtual, consulte el estado de los clústeres virtuales mediante la operación `list-virtual-clusters` o vaya a la página Clústeres virtuales de la consola de Amazon EMR.

Enviar una ejecución de trabajo con **StartJobRun**

Enviar una ejecución de trabajo con un archivo JSON con los parámetros especificados

1. Cree un archivo `start-job-run-request.json` y especifique los parámetros necesarios para la ejecución del trabajo, tal como se muestra en el siguiente archivo JSON de ejemplo. Para obtener más información sobre los parámetros, consulte [Opciones para configurar una ejecución de trabajo](#).

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.2.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
      "sparkSubmitParameters": "--class <main_class> --conf
spark.executor.instances=2 --conf spark.executor.memory=2G --conf
spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "2G"
        }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

2. Utilice el comando `start-job-run` con una ruta al archivo `start-job-run-request.json` almacenado localmente.

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

Para iniciar la ejecución de un trabajo con el comando **start-job-run**

1. Proporcione todos los parámetros especificados en el comando `StartJobRun`, tal como se muestra en el siguiente ejemplo.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
"entryPointArguments": ["argument1", "argument2", ...], "sparkSubmitParameters":
"--class <main_class> --conf spark.executor.instances=2 --conf
spark.executor.memory=2G --conf spark.executor.cores=2 --conf
spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
"spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
"monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
{"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
"persistentAppUI": "ENABLED", "s3MonitoringConfiguration": {"logUri":
"s3://my_s3_log_location" }}}
```

2. En el caso de Spark SQL, ingrese todos los parámetros especificados en el comando `StartJobRun`, tal y como se muestra en el siguiente ejemplo.

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

Uso de la clasificación de remitentes de trabajos

Descripción general

La solicitud de `StartJobRun` de Amazon EMR en EKS crea un pod de remitente de trabajos (también conocido como job-runner) para generar el controlador de Spark. Puede usar la `emr-job-submitter` clasificación para configurar los selectores de nodos para su módulo de envío de trabajos, así como configurar la imagen, la CPU y la memoria para el contenedor de registro del módulo de envío de trabajos.

Los siguientes ajustes están disponibles en la clasificación: `emr-job-submitter`

`jobsubmitter.node.selector.[labelKey]`

Agrega al selector de nodos del pod de remitente de trabajos, con la clave *labelKey* y el valor como el valor de configuración para la configuración. Por ejemplo, puede establecer `jobsubmitter.node.selector.identifier` en `myIdentifier` y el pod de remitentes de trabajos tendrá un valor de identificador clave de `myIdentifier`. Esto se puede usar para especificar en qué nodos se puede colocar el módulo de presentación de trabajos. Para agregar varias claves de selección de nodos, defina varias configuraciones con este prefijo.

`jobsubmitter.logging.image`

Establece una imagen personalizada que se utilizará en el contenedor de registro del módulo de envío de trabajos.

`jobsubmitter.logging.request.cores`

Establece un valor personalizado para el número CPUs, en unidades de CPU, del contenedor de registro del módulo de envío de trabajos. De forma predeterminada, está establecido en 100 m.

`jobsubmitter.logging.request.memory`

Establece un valor personalizado para la cantidad de memoria, en bytes, del contenedor de registro del módulo de envío de trabajos. De forma predeterminada, está establecido en 200 MB. Un mebibyte es una unidad de medida similar a un megabyte.

Recomendamos colocar los módulos de presentación de trabajos en las instancias bajo demanda. Colocar los módulos de envío de trabajos en las instancias puntuales podría provocar un error en el trabajo si la instancia en la que se ejecuta el módulo de envío de trabajos está sujeta a una interrupción de la instancia puntual. También puede [colocar el pod de remitente de trabajos en una única zona de disponibilidad](#) o [utilizar cualquier etiqueta de Kubernetes que se aplique a los nodos](#).

Ejemplos de clasificación de remitentes de trabajos

En esta sección

- [Solicitud de StartJobRun con ubicación de nodos bajo demanda para el pod de remitente de trabajos](#)
- [Solicitud de StartJobRun con colocación de nodos Single-AZ para el pod de remitente de trabajos](#)
- [StartJobRunsolicitud con ubicación de tipo EC2 instancia Single-AZ y de tipo Amazon para el pod de remitente del trabajo](#)
- [StartJobRunsolicitud con imagen, CPU y memoria de contenedor de registro personalizados](#)

Solicitud de **StartJobRun** con ubicación de nodos bajo demanda para el pod de remitente de trabajos

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

```

    }
  ],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodel-selector-job-submitter.json

```

Solicitud de **StartJobRun** con colocación de nodos Single-AZ para el pod de remitente de trabajos

```

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

```

```

    "properties": {
      "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
Zone"
    }
  },
],
"monitoringConfiguration": {
  "cloudWatchMonitoringConfiguration": {
    "logGroupName": "/emr-containers/jobs",
    "logStreamNamePrefix": "demo"
  },
  "s3MonitoringConfiguration": {
    "logUri": "s3://joblogs"
  }
}
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodelocator-job-submitter-az.json

```

StartJobRunssolicitud con ubicación de tipo EC2 instancia Single-AZ y de tipo Amazon para el pod de remitente del trabajo

```

{
  "name": "spark-python-in-s3-nodelocator",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
      "sparkSubmitParameters": "--conf spark.driver.cores=5 --conf
spark.kubernetes.pyspark.pythonVersion=3 --conf spark.executor.memory=20G
--conf spark.driver.memory=15G --conf spark.executor.cores=6 --conf
spark.sql.shuffle.partitions=1000"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {

```



```

        "spark.dynamicAllocation.enabled": "false",
    },
    {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
Zone",
            "jobsubmitter.node.selector.node.kubernetes.io/instance-type": "m5.4xlarge"
        }
    }
],
"monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
    }
}
}
}

```

StartJobRunsolicitud con imagen, CPU y memoria de contenedor de registro personalizados

```

{
    "name": "spark-python",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.11.0-latest",
    "jobDriver": {
        "sparkSubmitJobDriver": {
            "entryPoint": "s3://S3-prefix/trip-count.py"
        }
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "emr-job-submitter",
                "properties": {
                    "jobsubmitter.logging.image": "YOUR_ECR_IMAGE_URL",
                    "jobsubmitter.logging.request.memory": "200Mi",

```

```
        "jobsubmitter.logging.request.cores": "0.5"
      }
    }
  ],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
    }
  }
}
```

Uso de la clasificación por defecto del contenedor Amazon EMR

Descripción general

Los siguientes ajustes están disponibles en la `emr-containers-defaults` clasificación:

job-start-timeout

De forma predeterminada, se agota el tiempo de espera de un trabajo si no se puede iniciar y permanece en ese SUBMITTED estado durante 15 minutos. Esta configuración cambia el número de segundos que hay que esperar hasta que se agote el tiempo de espera del trabajo.

logging.image

Establece una imagen personalizada que se utilizará en el contenedor de registro de los módulos del controlador y del ejecutor.

logging.request.cores

Establece un valor personalizado para el número CPUs, en unidades de CPU, del contenedor de registro de los módulos del controlador y del ejecutor. De forma predeterminada, esto no está establecido.

logging.request.memory

Establece un valor personalizado para la cantidad de memoria, en bytes, del contenedor de registro de los pods del controlador y del ejecutor. De forma predeterminada, está establecido en 512Mi. Un mebibyte es una unidad de medida similar a un megabyte.

Ejemplos de clasificación de remitentes de trabajos

En esta sección

- [StartJobRunsolicitud con tiempo de espera de trabajo personalizado](#)
- [StartJobRunsolicitud con imagen, CPU y memoria de contenedor de registro personalizados](#)

StartJobRunsolicitud con tiempo de espera de trabajo personalizado

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-containers-defaults",
        "properties": {
          "job-start-timeout": "1800"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

StartJobRunsolicitud con imagen, CPU y memoria de contenedor de registro personalizados

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-containers-defaults",
        "properties": {
          "logging.image": "YOUR_ECR_IMAGE_URL",
          "logging.request.memory": "200Mi",
          "logging.request.cores": "0.5"
        }
      }
    ]
  },
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
    }
  }
}
```

Ejecución de trabajos de Spark con el operador de Spark

Las versiones 6.10.0 y posteriores de Amazon EMR admiten el operador de Kubernetes para Apache Spark, o el operador de Spark, como modelo de envío de trabajos para Amazon EMR en EKS. Con el operador de Spark, puede implementar y administrar las aplicaciones de Spark con el tiempo de ejecución de versiones de Amazon EMR en sus propios clústeres de Amazon EKS. Una vez que haya implementado el operador de Spark en su clúster de Amazon EKS, puede enviar las solicitudes

de Spark de inmediato al operador. El operador administra el ciclo de vida de las aplicaciones de Spark.

 Note

Amazon EMR calcula los precios de Amazon EKS en función de la vCPU y el consumo de memoria. Este cálculo se aplica a los pods de controladores y ejecutores. Este cálculo comienza cuando usted descarga la imagen de la aplicación de Amazon EMR hasta que finaliza el pod de Amazon EKS y se redondea al segundo más cercano.

Temas

- [Configuración del operador de Spark para Amazon EMR en EKS](#)
- [Cómo comenzar a utilizar el operador de Spark para Amazon EMR en EKS](#)
- [Uso del escalado automático vertical con el operador de Spark para Amazon EMR en EKS](#)
- [Desinstalación del operador de Spark para Amazon EMR en EKS](#)
- [Uso de la configuración de supervisión para supervisar el operador de Spark Kubernetes y los trabajos de Spark](#)
- [Seguridad y operador de Spark con Amazon EMR en EKS](#)

Configuración del operador de Spark para Amazon EMR en EKS

Complete las siguientes tareas para preparar la configuración antes de instalar el operador de Spark en Amazon EKS. Si ya se registró en Amazon Web Services (AWS) y ha usado Amazon EKS, lo tiene todo casi listo para comenzar a utilizar Amazon EMR en EKS. Complete las siguientes tareas para la configuración del operador de Spark en Amazon ECS. Si ya ha completado alguno de los requisitos previos, puede omitirlos y pasar al siguiente.

- [Instalar o actualizar a la última versión de AWS CLI](#): si ya ha instalado el AWS CLI, confirme que tiene la última versión.
- [Configure kubectl y eksctl](#) : eksctl es una herramienta de línea de comandos que se utiliza para comunicarse con Amazon EKS.
- [Instale Helm](#): el administrador de paquetes Helm para Kubernetes le ayuda a instalar y administrar aplicaciones en el clúster de Kubernetes.

- [Comience con Amazon EKS: eksctl](#) : siga los pasos para crear un nuevo clúster de Kubernetes con nodos en Amazon EKS.
- [Seleccione un URI de imagen base de Amazon EMR](#) (versión 6.10.0 o posterior): el operador de Spark es compatible con las versiones 6.10.0 y posteriores de Amazon EMR.

Cómo comenzar a utilizar el operador de Spark para Amazon EMR en EKS

Este tema le ayuda a comenzar a utilizar el operador de Spark en Amazon EKS mediante la implementación de una aplicación de Spark y una aplicación de Schedule Spark.

Instalar el operador de Spark

Siga estos pasos para instalar el operador de Kubernetes para Apache Spark.

1. Si aún no lo ha hecho, complete los pasos de [Configuración del operador de Spark para Amazon EMR en EKS](#).
2. Autentique su cliente de Helm en el registro de Amazon ECR. En el siguiente comando, sustituya los *region-id* valores de la [Cuentas de registro de Amazon ECR por región](#) página por los que prefiera Región de AWS y por el *ECR-registry-account* valor correspondiente a la región.

```
aws ecr get-login-password \  
--region region-id | helm registry login \  
--username AWS \  
--password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
```

3. Instale el operador de Spark con el siguiente comando.

Para el parámetro `--version` del gráfico de Helm, use su etiqueta de lanzamiento de Amazon EMR sin el prefijo `emr-` y sin el sufijo de fecha. Por ejemplo, con la versión `emr-6.12.0-java17-latest`, especifique `6.12.0-java17`. El ejemplo del siguiente comando usa la versión `emr-7.8.0-latest`, por lo que especifica `7.8.0` para el gráfico de Helm `--version`.

```
helm install spark-operator-demo \  
oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \  
--set emrContainers.awsRegion=region-id \  
--version 7.8.0 \  
--namespace spark-operator \  
--create-namespace
```

De forma predeterminada, el comando crea una cuenta de servicio `emr-containers-sa-spark-operator` para el operador de Spark. Para usar una cuenta de servicio diferente, proporcione el argumento `serviceAccounts.sparkoperator.name`. Por ejemplo:

```
--set serviceAccounts.sparkoperator.name my-service-account-for-spark-operator
```

Si quiere [usar el escalado automático vertical con el operador de Spark](#), agregue la siguiente línea al comando de instalación para admitir webhooks para el operador:

```
--set webhook.enable=true
```

4. Compruebe que haya instalado el gráfico de Helm con el comando `helm list`:

```
helm list --namespace spark-operator -o yaml
```

El comando `helm list` debería devolver la información de lanzamiento del gráfico de Helm recién implementado:

```
app_version: v1beta2-1.3.8-3.1.1
chart: spark-operator-7.8.0
name: spark-operator-demo
namespace: spark-operator
revision: "1"
status: deployed
updated: 2023-03-14 18:20:02.721638196 +0000 UTC
```

5. Complete la instalación con todas las opciones adicionales que necesite. Para obtener más información, consulte la [spark-on-k8s-operator](#) documentación en GitHub.

Ejecutar una aplicación de Spark

El operador de Spark es compatible con Amazon EMR 6.10.0 o una versión posterior. Cuando instala el operador de Spark, este crea la cuenta de servicio `emr-containers-sa-spark` para ejecutar las aplicaciones de Spark de forma predeterminada. Siga estos pasos para ejecutar una aplicación de Spark con el operador de Spark en Amazon EMR en EKS 6.10.0 o una versión posterior.

1. Antes de poder ejecutar una aplicación de Spark con el operador de Spark, complete los pasos indicados en [Configuración del operador de Spark para Amazon EMR en EKS](#) y [Instalar el operador de Spark](#).
2. Cree un archivo de definición de SparkApplication `spark-pi.yaml` con el siguiente contenido:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
```



```
volumeMounts:
  - name: "test-volume"
    mountPath: "/tmp"
```

3. Luego, implemente la aplicación de Spark con el siguiente comando. Esto también creará un objeto SparkApplication denominado spark-pi:

```
kubectl apply -f spark-pi.yaml
```

4. Compruebe los eventos del objeto SparkApplication con el siguiente comando:

```
kubectl describe sparkapplication spark-pi --namespace spark-operator
```

Para obtener más información sobre cómo enviar solicitudes a Spark a través del operador de Spark, consulte [Cómo usar un SparkApplication](#) en la spark-on-k8s-operator documentación de GitHub.

Uso de Amazon S3 para almacenamiento

Para usar Amazon S3 como opción de almacenamiento de archivos, añada las siguientes configuraciones al archivo YAML.

```
hadoopConf:
# EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
    com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
  mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
    "2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
# Required for EMR Runtime
  spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
```

```
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

Si utiliza las versiones 7.2.0 o posteriores de Amazon EMR, las configuraciones se incluyen de forma predeterminada. En ese caso, puede establecer la ruta del archivo en `s3://<bucket_name>/<file_path>` en lugar de en `local://<file_path>` en el archivo YAML de la aplicación Spark.

A continuación, envíe la aplicación de Spark como de costumbre.

Uso del escalado automático vertical con el operador de Spark para Amazon EMR en EKS

A partir de Amazon EMR 7.0, puede usar Amazon EMR en el escalado automático vertical de EKS para simplificar la administración de recursos. Ajusta automáticamente los recursos de memoria y CPU para adaptarlos a las necesidades de la carga de trabajo que proporciona a las aplicaciones de Spark de Amazon EMR. Para obtener más información, consulte [Uso del escalado automático vertical con trabajos de Spark de Amazon EMR](#).

En esta sección, se describe cómo configurar el operador de Spark para usar el escalado automático vertical.

Requisitos previos

Antes de configurar la supervisión, asegúrese de completar las siguientes tareas de configuración:

- Realice los pasos que se indican en [Configuración del operador de Spark para Amazon EMR en EKS](#).
- (opcional) Si ya instaló una versión anterior del operador Spark, elimine la SparkApplication ScheduledSparkApplication /CRD.

```
kubectl delete crd sparkApplication
kubectl delete crd scheduledSparkApplication
```

- Realice los pasos que se indican en [Instalar el operador de Spark](#). En el paso 3, agregue la siguiente línea al comando de instalación para admitir webhooks para el operador:

```
--set webhook.enable=true
```

- Realice los pasos que se indican en [Configuración del escalado automático vertical de Amazon EMR en EKS](#).
- Conceda acceso a los archivos de su ubicación de Amazon S3:

1. Anote su cuenta de servicio del controlador y el operador con el JobExecutionRole que tenga permisos de S3.

```
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark
eks.amazonaws.com/role-arn=JobExecutionRole
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark-
operator eks.amazonaws.com/role-arn=JobExecutionRole
```

2. Actualice la política de confianza de su rol de ejecución de trabajos en ese espacio de nombres.

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace ${Namespace} \
--role-name iam_role_name_for_job_execution
```

3. Edite la política de confianza del rol de IAM de su rol de ejecución de trabajos y actualice la serviceaccount de emr-containers-sa-spark-*-* -xxxx a emr-containers-sa-*.


```
{
  "Effect": "Allow",
  "Principal": {
    "Federated": "OIDC-provider"
  },
  "Action": "sts:AssumeRoleWithWebIdentity",
  "Condition": {
    "StringLike": {
```

```
    "StringLike": {
```

```

        "OIDC": "system:serviceaccount:${Namespace}:emr-containers-sa-*"
    }
}
}

```

4. Si utiliza Amazon S3 como almacenamiento de archivos, añada los valores predeterminados siguientes a su archivo yaml.

```

hadoopConf:
# EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
    com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

  mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
    "2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
  # Required for EMR Runtime
  spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/
aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
  spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
  spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
  spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native

```

Ejecutar un trabajo con escalado automático vertical en el operador de Spark

Para poder ejecutar una aplicación de Spark con el operador de Spark, complete los pasos indicados en [Requisitos previos](#).

Para utilizar el escalado automático vertical con el operador de Spark, añada la siguiente configuración al controlador para la especificación de la aplicación de Spark a fin de activar el escalado automático vertical:

```
dynamicSizing:
  mode: Off
  signature: "my-signature"
```

Esta configuración permite el escalado automático vertical y es una configuración de firma obligatoria que le permite elegir una firma para su trabajo.

Para obtener más información sobre las configuraciones y los valores de los parámetros, consulte [Configuración del escalado automático vertical para Amazon EMR en EKS](#). De forma predeterminada, su trabajo se envía en el modo Desactivado de escalado automático vertical solo de supervisión. Este estado de supervisión le permite calcular y ver las recomendaciones de recursos sin llevar a cabo el escalado automático. Para obtener más información, consulte [Modos de escalado automático](#).

El siguiente ejemplo muestra un archivo de definición de SparkApplication denominado spark-pi.yaml y con las configuraciones necesarias para utilizar el escalado automático vertical.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.8.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.4.1"
  dynamicSizing:
    mode: Off
    signature: "my-signature"
```

```
restartPolicy:
  type: Never
volumes:
  - name: "test-volume"
    hostPath:
      path: "/tmp"
      type: Directory
driver:
  cores: 1
  coreLimit: "1200m"
  memory: "512m"
  labels:
    version: 3.4.1
  serviceAccount: emr-containers-sa-spark
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
executor:
  cores: 1
  instances: 1
  memory: "512m"
  labels:
    version: 3.4.1
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
```

Luego, implemente la aplicación de Spark con el siguiente comando. Esto también creará un objeto `SparkApplication` denominado `spark-pi`:

```
kubectl apply -f spark-pi.yaml
```

Para obtener más información sobre cómo enviar solicitudes a Spark a través del operador de Spark, consulte [Cómo usar un SparkApplication](#) en la `spark-on-k8s-operator` documentación de GitHub.

Verificación de la funcionalidad de escalado automático vertical

Para comprobar que el escalado automático vertical funcione correctamente en el trabajo enviado, use `kubectl` para obtener el recurso personalizado `verticalpodautoscaler` y ver sus recomendaciones de escalado.

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=my-signature
```

El resultado de esta consulta debe parecerse al siguiente:

NAMESPACE	NAME	MODE		
CPU	MEM	PROVIDED	AGE	
spark-operator	ds-p73j6mkosvc4xeb3gr7x4xol2bfcw5evqimzqojrlysvj3giozuq-vpa	Off		
580026651	True	15m		

Si el resultado no es similar o contiene un código de error, consulte [Solución de problemas en el escalado automático vertical de Amazon EMR en EKS](#) para ver los pasos que le ayudarán a resolver el problema.

Para eliminar los pods y las aplicaciones del catálogo, ejecute el siguiente comando:

```
kubectl delete sparkapplication spark-pi
```

Desinstalación del operador de Spark para Amazon EMR en EKS

Siga estos pasos para desinstalar el operador de Spark.

1. Elimine el operador de Spark con el espacio de nombres correcto. En este ejemplo, el espacio de nombres es `spark-operator-demo`.

```
helm uninstall spark-operator-demo -n spark-operator
```

2. Elimine la cuenta de servicio del operador de Spark:

```
kubectl delete sa emr-containers-sa-spark-operator -n spark-operator
```

3. Elimina el operador Spark CustomResourceDefinitions (CRDs):

```
kubectl delete crd sparkapplications.sparkoperator.k8s.io
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
```

Uso de la configuración de supervisión para supervisar el operador de Spark Kubernetes y los trabajos de Spark

La configuración de supervisión le permite configurar fácilmente el archivado de registros de sus aplicaciones y registros de operadores de Spark en Amazon S3 o Amazon CloudWatch. Puede elegir uno o ambos. Al hacerlo, se agrega un contenedor asociado a las cápsulas de Spark, al controlador y al ejecutor de Spark y, posteriormente, se reenvían los registros de estos componentes a los receptores configurados.

Requisitos previos

Antes de configurar la supervisión, asegúrese de completar las siguientes tareas de configuración:

1. (Opcional) Si ya instaló una versión anterior del operador Spark, elimine la SparkApplicationScheduledSparkApplication/CRD.

```
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
kubectl delete crd sparkapplications.sparkoperator.k8s.io
```

2. Cree un rol de operador/de ejecución de tareas en IAM si aún no lo tiene.
3. Para actualizar la política de confianza del rol de operador o ejecución de tareas que acaba de crear, ejecute el siguiente comando:

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace namespace \
--role-name iam_role_name_for_operator/job_execution_role
```

4. Edite la política de confianza del rol de IAM de su rol de operador/de ejecución de tareas de la siguiente manera:

```
{
  "Effect": "Allow",
  "Principal": {
    "Federated": "${OIDC-provider}"
  },
  "Action": "sts:AssumeRoleWithWebIdentity",
  "Condition": {
    "StringLike": {
```



```

        "OIDC_PROVIDER:sub": "system:serviceaccount:${Namespace}:emr-
containers-sa-*"
      }
    }
  }
}

```

5. Cree una política de monitorización y configuración en IAM con los siguientes permisos:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:region:account_id:log-group:log_group_name",
        "arn:aws:logs:region:account_id:log-group:log_group_name:"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "logs:DescribeLogGroups",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::bucket_name",
        "arn:aws:s3:::bucket_name/*"
      ]
    }
  ]
}

```

6. Adjunte la política anterior a su función de operador/de ejecución de tareas.

Registros de operadores de Spark

Puede definir la configuración de supervisión de la siguiente manera `helm install`:

```
helm install spark-operator spark-operator \
--namespace namespace \
--set emrContainers.awsRegion=aws_region \
--set emrContainers.monitoringConfiguration.image=log_agent_image_url \
--set
  emrContainers.monitoringConfiguration.s3MonitoringConfiguration.logUri=S3_bucket_uri \
--set
  emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logGroupName=log_group
\
--set
  emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logStreamNamePrefix=log_stream_name_prefix
\
--set emrContainers.monitoringConfiguration.sideCarResources.limits.cpuLimit=500m \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.memoryLimit=512Mi \
--set
  emrContainers.monitoringConfiguration.containerLogRotationConfiguration.rotationSize=2GB
\
--set
  emrContainers.monitoringConfiguration.containerLogRotationConfiguration.maxFilesToKeep=10
\
--set webhook.enable=true \
--set emrContainers.operatorExecutionRoleArn=operator_execution_role_arn
```

Configuración de monitoreo

A continuación se describen las opciones de configuración disponibles de `monitoringConfiguration`.

- Imagen (opcional): URL de la imagen del agente de registro. Se recuperará `emrReleaseLabel` si no se proporciona.
- `s3 MonitoringConfiguration`: configure esta opción para archivar en Amazon S3.
 - `logUri`: (obligatorio): la ruta del bucket de Amazon S3 donde desea almacenar sus registros.
 - Los siguientes son ejemplos de formatos para las rutas de bucket de Amazon S3 después de cargar los registros. El primer ejemplo muestra que la rotación de registros no está habilitada.

```
s3://${logUri}/${POD_NAME}/operator/stdout.gz
```

```
s3://${logUri}/${POD_NAME}/operator/stderr.gz
```

La rotación de registros está habilitada de forma predeterminada. Puede ver tanto un archivo girado, con un índice creciente, como un archivo actual, que es el mismo que en el ejemplo anterior.

```
s3://${logUri}/${POD_NAME}/operator/stdout_YYYYMMDD_index.gz  
s3://${logUri}/${POD_NAME}/operator/stderr_YYYYMMDD_index.gz
```

- **cloudWatchMonitoringConfiguración:** la clave de configuración a la que se debe configurar el reenvío. Amazon CloudWatch
 - **logGroupName(obligatorio):** nombre del grupo de Amazon CloudWatch registros al que quiere enviar registros. Si el grupo no existe, se crea automáticamente.
 - **logStreamNameprefijo (opcional):** nombre del flujo de registro al que quiere enviar registros. El valor predeterminado es una cadena vacía. El formato Amazon CloudWatch es el siguiente:

```
${logStreamNamePrefix}/${POD_NAME}/STDOUT or STDERR
```

- **sideCarResources(opcional):** la clave de configuración para establecer los límites de recursos en el contenedor asociado de Fluentd lanzado.
 - **memoryLimit (opcional):** el límite de memoria. Ajústelo según sus necesidades. El valor predeterminado es 512Mi.
 - **cpuLimit (opcional):** el límite de la CPU. Ajústelo según sus necesidades. El valor predeterminado es 500 m.
- **containerLogRotationConfiguración (opcional):** controla el comportamiento de la rotación del registro de contenedor. Está habilitada de forma predeterminada.
 - **rotationSize (obligatorio):** especifica el tamaño del archivo para la rotación del registro. El rango de valores posibles va de 2 KB a 2 GB. La parte de la unidad numérica del parámetro rotationSize se pasa como un número entero. Como no se admiten valores decimales, puede especificar un tamaño de rotación de 1,5 GB, por ejemplo, con el valor 1500 MB. El valor predeterminado es 2 GB.
 - **maxFilesToconservar (obligatorio):** especifica el número máximo de archivos que deben retenerse en el contenedor una vez hecha la rotación. El valor mínimo es 1 y el máximo, 50. El valor predeterminado es 10.

Tras configurar MonitoringConfiguration, deberías poder comprobar los registros de los pods de Spark Operator en un bucket de Amazon S3 o Amazon CloudWatch en ambos. Para un bucket de Amazon S3, debe esperar 2 minutos para que se vacíe el primer archivo de registro.

Para encontrar los registros Amazon CloudWatch, vaya a lo siguiente: CloudWatch> Grupos de registros > > /operator/stderr **Log group namePod name**

También puede ir a: > Grupos de registros > > /operator/stdout CloudWatch**Log group namePod name**

Registros de aplicaciones de Spark

Puede definir la configuración de la siguiente manera:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: namespace
spec:
  type: Scala
  mode: cluster
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  emrReleaseLabel: emr_release_label
  executionRoleArn: job_execution_role_arn
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
```

```

    mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  monitoringConfiguration:
    image: "log_agent_image"
    s3MonitoringConfiguration:
      logUri: "S3_bucket_uri"
    cloudWatchMonitoringConfiguration:
      logGroupName: "log_group_name"
      logStreamNamePrefix: "log_stream_prefix"
  sidecarResources:
    limits:
      cpuLimit: "500m"
      memoryLimit: "250Mi"
  containerLogRotationConfiguration:
    rotationSize: "2GB"
    maxFilesToKeep: "10"

```

A continuación se describen las opciones de configuración disponibles de `monitoringConfiguration`.

- Imagen (opcional): URL de la imagen del agente de registro. Se recuperará `emrReleaseLabel` si no se proporciona.
- `s3 MonitoringConfiguration`: configure esta opción para archivar en Amazon S3.
- `logUri` (obligatorio): la ruta del bucket de Amazon S3 donde desea almacenar sus registros. El primer ejemplo muestra que la rotación de registros no está habilitada:

```

s3://${logUri}/${APPLICATION_NAME}-${APPLICATION_UID}/${POD_NAME}/stdout.gz
s3://${logUri}/${APPLICATION_NAME}-${APPLICATION_UID}/${POD_NAME}/stderr.gz

```

La rotación de registros no está habilitada de forma predeterminada. Puede utilizar tanto un archivo rotado (con índice creciente) como un archivo actual (uno sin la fecha).

```

s3://${logUri}/${APPLICATION_NAME}-${APPLICATION_UID}/${POD_NAME}/
stdout_YYYYMMDD_index.gz

```

```
s3://${logUri}/${APPLICATION_NAME}-${APPLICATION_UID}/${POD_NAME}/
stderr_YYYYMMDD_index.gz
```

- **cloudWatchMonitoringConfiguración:** la clave de configuración a la que se debe configurar el reenvío. Amazon CloudWatch
- **logGroupName(obligatorio):** el nombre del grupo de registros de Cloudwatch al que quiere enviar registros. Si el grupo no existe, se crea automáticamente.
- **logStreamNameprefijo (opcional):** el nombre del flujo de registro al que quiere enviar registros. El valor predeterminado es una cadena vacía. El formato CloudWatch es el siguiente:

```
${logStreamNamePrefix}/${APPLICATION_NAME}-${APPLICATION_UID}/${POD_NAME}/stdout
${logStreamNamePrefix}/${APPLICATION_NAME}-${APPLICATION_UID}/${POD_NAME}/stderr
```

- **sideCarResources(opcional):** la clave de configuración para establecer los límites de recursos en el contenedor asociado de Fluentd lanzado.
 - **memoryLimit (opcional):** el límite de memoria. Ajústelo según sus necesidades. El valor predeterminado es 250 Mi.
 - **cpuLimit:** el límite de la CPU. Ajústelo según sus necesidades. El valor predeterminado es 500 m.
- **containerLogRotationConfiguración (opcional):** controla el comportamiento de la rotación del registro de contenedor. Está habilitada de forma predeterminada.
 - **rotationSize (obligatorio):** especifica el tamaño del archivo para la rotación del registro. El rango de valores posibles va de 2 KB a 2 GB. La parte de la unidad numérica del parámetro rotationSize se pasa como un número entero. Como no se admiten valores decimales, puede especificar un tamaño de rotación de 1,5 GB, por ejemplo, con el valor 1500 MB. El valor predeterminado es 2 GB.
 - **maxFilesToconservar (obligatorio):** especifica el número máximo de archivos que deben retenerse en el contenedor una vez hecha la rotación. El valor mínimo es 1. El valor máximo es 50. El valor predeterminado es 10.

Tras configurar MonitoringConfiguration, deberías poder comprobar los registros del controlador y del ejecutor de la aplicación Spark en un bucket de Amazon S3 o en ambos. CloudWatch Para un bucket de Amazon S3, debe esperar 2 minutos para que se vacíe el primer archivo de registro. Por ejemplo, en Amazon S3, la ruta del bucket tiene el siguiente aspecto:

Amazon S3 > Buckets > **Bucket name** > **Spark application name - UUID Pod Name** > stderr.gz

O bien:

Amazon S3 > Buckets > **Bucket name** > **Spark application name - UUID Pod Name** > stdout.gz

En CloudWatch, la ruta aparece de la siguiente manera:

CloudWatch> Grupos de registros > **Log group name** > **Spark application name - UUID/Pod name**/stderr

O bien:

CloudWatch> Grupos de registros > >/**Log group name**/stdout **Spark application name - UUID Pod name**

Seguridad y operador de Spark con Amazon EMR en EKS

Existen dos formas de configurar los permisos de acceso a los clústeres cuando se utiliza el operador Spark. La primera es utilizar el control de acceso basado en roles, el control de acceso basado en roles (RBAC) restringe el acceso en función del rol de una persona dentro de una organización. Se ha convertido en la principal forma de gestionar el acceso. El segundo método de acceso consiste en asumir un AWS Identity and Access Management rol, que proporciona acceso a los recursos mediante permisos específicos asignados.

Temas

- [Configuración de los permisos de acceso al clúster con control de acceso basado en roles \(RBAC\)](#)
- [Configuración de los permisos de acceso al clúster con roles de IAM para las cuentas de servicio \(IRSA\)](#)

Configuración de los permisos de acceso al clúster con control de acceso basado en roles (RBAC)

Para implementar el operador de Spark, Amazon EMR en EKS crea dos roles y cuentas de servicio para el operador y las aplicaciones de Spark.

Temas

- [Rol y cuenta de servicio del operador](#)
- [Rol y cuenta de servicio de Spark](#)

Rol y cuenta de servicio del operador

Amazon EMR en EKS crea el rol y la cuenta de servicio del operador para administrar SparkApplications para los trabajos de Spark y otros recursos, como los servicios.

El nombre predeterminado de esta cuenta de servicio es `emr-containers-sa-spark-operator`.

Las siguientes reglas se aplican a este rol de servicio:

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  - configmaps
  - secrets
  verbs:
  - create
  - get
  - delete
  - update
- apiGroups:
  - extensions
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - create
  - get
  - delete
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
```



```
resources:
- events
verbs:
- create
- update
- patch
- apiGroups:
- ""
resources:
- resourcequotas
verbs:
- get
- list
- watch
- apiGroups:
- apiextensions.k8s.io
resources:
- customresourcedefinitions
verbs:
- create
- get
- update
- delete
- apiGroups:
- admissionregistration.k8s.io
resources:
- mutatingwebhookconfigurations
- validatingwebhookconfigurations
verbs:
- create
- get
- update
- delete
- apiGroups:
- sparkoperator.k8s.io
resources:
- sparkapplications
- sparkapplications/status
- scheduledsparkapplications
- scheduledsparkapplications/status
verbs:
- "*"
{{- if .Values.batchScheduler.enable }}
# required for the `volcano` batch scheduler
```

```
- apiGroups:
  - scheduling.incubator.k8s.io
  - scheduling.sigs.dev
  - scheduling.volcano.sh
resources:
  - podgroups
verbs:
  - "*"
{{- end }}
{{ if .Values.webhook.enable }}
- apiGroups:
  - batch
resources:
  - jobs
verbs:
  - delete
{{- end }}
```

Rol y cuenta de servicio de Spark

Un pod controlador de Spark necesita una cuenta de servicio de Kubernetes en el mismo espacio de nombres que el pod. Esta cuenta de servicio necesita permisos para crear, obtener, enumerar, parchear y eliminar los pods ejecutores, así como para crear un servicio Headless de Kubernetes para el controlador. El controlador produce un error y se cierra sin la cuenta de servicio, a menos que la cuenta de servicio predeterminada del espacio de nombres del pod tenga los permisos necesarios.

El nombre predeterminado de esta cuenta de servicio es `emr-containers-sa-spark`.

Las siguientes reglas se aplican a este rol de servicio:

```
rules:
- apiGroups:
  - ""
resources:
  - pods
verbs:
  - "*"
- apiGroups:
  - ""
resources:
  - services
verbs:
  - "*"

```

```
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"

```

Configuración de los permisos de acceso al clúster con roles de IAM para las cuentas de servicio (IRSA)

En esta sección se utiliza un ejemplo para demostrar cómo configurar una cuenta de servicio de Kubernetes para que asuma un rol. AWS Identity and Access Management. Los pods que usan la cuenta de servicio pueden acceder a cualquier AWS servicio de al que el rol tenga permisos para acceder.

En el siguiente ejemplo, se ejecuta una aplicación de Spark para contar las palabras de un archivo en Amazon S3. Para ello, puede configurar roles de IAM para las cuentas de servicio (IRSA) a fin de autenticar y autorizar las cuentas de servicio de Kubernetes.

Note

En este ejemplo, se utiliza el espacio de nombres “spark-operator” para el operador de Spark y para el espacio de nombres al que se envía la solicitud de Spark.

Requisitos previos

Antes de probar el ejemplo de esta página, complete los siguientes requisitos previos:

- [Prepare la configuración para el operador de Spark.](#)
- [Instalar el operador de Spark.](#)
- [Cree un bucket de Amazon S3.](#)
- Guarde su poema favorito en un archivo de texto llamado poem.txt y súbalo a su bucket de S3. La aplicación de Spark que cree en esta página leerá el contenido del archivo de texto. Para

obtener más información sobre cómo cargar archivos en S3, consulte [Cargar un objeto en el bucket](#) en la Guía del usuario de Amazon Simple Storage Service.

Configurar una cuenta de servicio de Kubernetes para que asuma un rol de IAM

Siga estos pasos para configurar una cuenta de servicio de Kubernetes a fin de que asuma un rol de IAM que los pods puedan usar para acceder a los AWS servicios de a los que el rol tiene permisos para acceder.

1. Tras completar [Requisitos previos](#), utilice AWS Command Line Interface para crear un `example-policy.json` archivo que permita el acceso de solo lectura al archivo que ha subido a Amazon S3:

```
cat >example-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-pod-bucket",
        "arn:aws:s3:::my-pod-bucket/*"
      ]
    }
  ]
}
EOF
```

2. Luego, cree una política de IAM `example-policy`:

```
aws iam create-policy --policy-name example-policy --policy-document file://
example-policy.json
```

3. A continuación, cree un rol de IAM `example-role` y asócielo a una cuenta de servicio de Kubernetes para el controlador de Spark:

```
eksctl create iamserviceaccount --name driver-account-sa --namespace spark-operator \
--cluster my-cluster --role-name "example-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/example-policy --approve
```

4. Cree un archivo yaml con las vinculaciones de roles de clúster necesarias para la cuenta de servicio de controlador de Spark:

```
cat >spark-rbac.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: driver-account-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: spark-role
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
- kind: ServiceAccount
  name: driver-account-sa
  namespace: spark-operator
EOF
```

5. Aplique las configuraciones de la vinculación de roles del clúster:

```
kubectl apply -f spark-rbac.yaml
```

El comando kubectl debería confirmar que la cuenta se ha creado correctamente:

```
serviceaccount/driver-account-sa created
clusterrolebinding.rbac.authorization.k8s.io/spark-role configured
```

Ejecutar una aplicación desde el operador de Spark

Después de [configurar la cuenta de servicio de Kubernetes](#), puede ejecutar una aplicación de Spark que cuente el número de palabras del archivo de texto que ha subido como parte de [Requisitos previos](#).

1. Cree un archivo nuevo `word-count.yaml` con una definición de `SparkApplication` para su aplicación de recuento de palabras.

```
cat >word-count.yaml <<EOF
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: word-count
  namespace: spark-operator
spec:
  type: Java
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.JavaWordCount
  mainApplicationFile: local:///usr/lib/spark/examples/jars/spark-examples.jar
  arguments:
    - s3://my-pod-bucket/poem.txt
  hadoopConf:
    # EMRFS filesystem
    fs.s3.customAWSCredentialsProvider:
com.amazonaws.auth.WebIdentityTokenCredentialsProvider
    fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
    fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
    fs.s3.buffer.dir: /mnt/s3
    fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
"2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
  sparkConf:
    # Required for EMR Runtime
    spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
```

```

glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
    spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
    sparkVersion: "3.3.1"
    restartPolicy:
        type: Never
    driver:
        cores: 1
        coreLimit: "1200m"
        memory: "512m"
        labels:
            version: 3.3.1
        serviceAccount: my-spark-driver-sa
    executor:
        cores: 1
        instances: 1
        memory: "512m"
        labels:
            version: 3.3.1
EOF

```

2. Envíe la aplicación de Spark.

```
kubectl apply -f word-count.yaml
```

El comando `kubectl` debería devolver la confirmación de que ha creado correctamente un objeto `SparkApplication` llamado `word-count`.

```
sparkapplication.sparkoperator.k8s.io/word-count configured
```

3. Para comprobar los eventos del objeto `SparkApplication`, ejecute el siguiente comando:

```
kubectl describe sparkapplication word-count -n spark-operator
```

El comando `kubectl` debería devolver la descripción de `SparkApplication` junto con los eventos:

```
Events:
  Type      Reason                                     Age          From
  Message
  ----      -
  Normal    SparkApplicationSpecUpdateProcessed      3m2s (x2 over 17h)    spark-
operator Successfully processed spec update for SparkApplication word-count
Warning    SparkApplicationPendingRerun             3m2s (x2 over 17h)    spark-
operator SparkApplication word-count is pending rerun
Normal     SparkApplicationSubmitted                 2m58s (x2 over 17h)    spark-
operator SparkApplication word-count was submitted successfully
Normal     SparkDriverRunning                       2m56s (x2 over 17h)    spark-
operator Driver word-count-driver is running
Normal     SparkExecutorPending                     2m50s                spark-
operator Executor [javawordcount-fdd1698807392c66-exec-1] is pending
Normal     SparkExecutorRunning                     2m48s                spark-
operator Executor [javawordcount-fdd1698807392c66-exec-1] is running
Normal     SparkDriverCompleted                     2m31s (x2 over 17h)    spark-
operator Driver word-count-driver completed
Normal     SparkApplicationCompleted                 2m31s (x2 over 17h)    spark-
operator SparkApplication word-count completed
Normal     SparkExecutorCompleted                   2m31s (x2 over 2m31s)  spark-
operator Executor [javawordcount-fdd1698807392c66-exec-1] completed
```

La aplicación ahora cuenta las palabras del archivo de S3. Para saber el número de palabras, consulte los archivos de registro de su controlador:

```
kubectl logs pod/word-count-driver -n spark-operator
```

El comando `kubectl` debería devolver el contenido del archivo de registro con los resultados de la aplicación de recuento de palabras.

```
INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 5.146519 s
Software: 1
```


[Para obtener más información sobre cómo enviar solicitudes a Spark a través del operador de Kubernetes para Apache Spark \(spark-on-k8s-operator\) SparkApplication en. GitHub](#)

Ejecución de trabajos de Spark con spark-submit

Las versiones 6.10.0 y posteriores de Amazon EMR admiten `spark-submit` como herramienta de línea de comandos que puede utilizar para enviar y ejecutar aplicaciones de Spark en un clúster de Amazon EMR en EKS.

Note

Amazon EMR calcula los precios de Amazon EKS en función de la vCPU y el consumo de memoria. Este cálculo se aplica a los pods de controladores y ejecutores. Este cálculo comienza cuando usted descarga la imagen de la aplicación de Amazon EMR hasta que finaliza el pod de Amazon EKS y se redondea al segundo más cercano.

Temas

- [Configuración de spark-submit para Amazon EMR en EKS](#)
- [Comenzar a utilizar spark-submit para Amazon EMR en EKS](#)
- [Verificación de los requisitos de seguridad de la cuenta de servicio del controlador de Spark para spark-submit](#)

Configuración de spark-submit para Amazon EMR en EKS

Complete las siguientes tareas para llevar a cabo la configuración antes de poder ejecutar una aplicación con `spark-submit` en Amazon EMR en EKS. Si ya se registró en Amazon Web Services (AWS) y ha usado Amazon EKS, lo tiene todo casi listo para comenzar a utilizar Amazon EMR en EKS. Si ya ha completado alguno de los requisitos previos, puede omitirlos y pasar al siguiente.

- [Instalar o actualizar a la última versión de AWS CLI](#): si ya ha instalado el AWS CLI, confirme que tiene la última versión.
- [Configure kubectl y eksctl](#): `eksctl` es una herramienta de línea de comandos que se utiliza para comunicarse con Amazon EKS.
- [Comience con Amazon EKS: eksctl](#): siga los pasos para crear un nuevo clúster de Kubernetes con nodos en Amazon EKS.

- [Seleccione un URI de imagen base de Amazon EMR](#) (versión 6.10.0 o posterior): el comando `spark-submit` es compatible con las versiones 6.10.0 y posteriores de Amazon EMR.
- Confirme que la cuenta de servicio de controlador tiene los permisos adecuados para crear y supervisar los módulos ejecutores. Para obtener más información, consulte [Verificación de los requisitos de seguridad de la cuenta de servicio del controlador de Spark para spark-submit](#).
- Configure su [perfil de credenciales de AWS](#) local.
- En la consola de Amazon EKS, elija su clúster de EKS y, a continuación, busque el punto de conexión del clúster de EKS, que se encuentra en Descripción general, Detalles y, a continuación, en el punto de conexión del servidor API.

Comenzar a utilizar spark-submit para Amazon EMR en EKS

Amazon EMR 6.10.0 y las versiones posteriores admiten `spark-submit` para ejecutar aplicaciones de Spark en un clúster de Amazon EKS. En la siguiente sección, se muestra cómo enviar un comando para una aplicación Spark.

Ejecutar una aplicación de Spark

Complete los pasos que se indican a continuación para ejecutar la aplicación de Spark:

1. Para poder ejecutar una aplicación de Spark con el comando `spark-submit`, complete los pasos que se indican en [Configuración de spark-submit para Amazon EMR en EKS](#).
2. Ejecute un contenedor con Amazon EMR en la imagen base de EKS. Consulte [Cómo seleccionar un URI de imagen base](#) para obtener más información.

```
kubectl run -it containerName --image=EMRonEKSIImage --command -n namespace /bin/  
bash
```

3. Establezca los valores de las siguientes variables de entorno:

```
export SPARK_HOME=spark-home  
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

4. Luego, envíe la solicitud de Spark con el siguiente comando:

```
$SPARK_HOME/bin/spark-submit \  
--class org.apache.spark.examples.SparkPi \  
--master $MASTER_URL \  

```

```
--conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest \  
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \  
--deploy-mode cluster \  
--conf spark.kubernetes.namespace=spark-operator \  
local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

Para obtener más información acerca de cómo enviar aplicaciones a Spark, consulte [Envío de aplicaciones](#) en la documentación de Apache Spark.

 Important

`spark-submit` solo admite el modo de clúster como mecanismo de envío.

Verificación de los requisitos de seguridad de la cuenta de servicio del controlador de Spark para `spark-submit`

El pod controlador de Spark utiliza una cuenta de servicio de Kubernetes para acceder al servidor de la API de Kubernetes y crear y supervisar los pods ejecutores. La cuenta de servicio del controlador debe tener los permisos adecuados para enumerar, crear, editar, parchear y eliminar los pods de su clúster. Puede verificar que puede enumerar estos recursos con el siguiente comando:

```
kubectl auth can-i list/create/edit/delete/patch pods
```

Verifique que tiene los permisos necesarios para ejecutar cada comando.

```
kubectl auth can-i list pods  
kubectl auth can-i create pods  
kubectl auth can-i edit pods  
kubectl auth can-i delete pods  
kubectl auth can-i patch pods
```

Las siguientes reglas se aplican a este rol de servicio:

```
rules:  
- apiGroups:  
  - ""
```

```
resources:
- pods
verbs:
- "*"
- apiGroups:
- ""
resources:
- services
verbs:
- "*"
- apiGroups:
- ""
resources:
- configmaps
verbs:
- "*"
- apiGroups:
- ""
resources:
- persistentvolumeclaims
verbs:
- "*"

```

Configuración de roles de IAM para cuentas de servicio (IRSA) para spark-submit

En las siguientes secciones, se explica cómo configurar roles de IAM para las cuentas de servicio (IRSA) a fin de autenticar y autorizar las cuentas de servicio de Kubernetes para que usted pueda ejecutar aplicaciones de Spark almacenadas en Amazon S3.

Requisitos previos

Antes de probar cualquiera de los ejemplos de esta documentación, asegúrese de que ha completado los siguientes requisitos previos:

- [Ha finalizado la configuración de spark-submit](#)
- [Ha creado un bucket de S3](#) y [ha cargado](#) el jar de aplicaciones de Spark

Configuración de una cuenta de servicio de Kubernetes para asumir un rol de IAM

En los siguientes pasos se explica cómo configurar una cuenta de servicio de Kubernetes para que asuma un rol de (IAM) AWS Identity and Access Management . Tras configurar los pods para que

usen la cuenta de servicio, podrán acceder a cualquier cuenta a la Servicio de AWS que el rol tenga permiso de acceso.

1. [Cree un archivo de políticas para conceder acceso de solo lectura al objeto de Amazon S3 que cargó:](#)

```
cat >my-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::<my-spark-jar-bucket>",
        "arn:aws:s3:::<my-spark-jar-bucket>/*"
      ]
    }
  ]
}
EOF
```

2. Creación de la política de IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

3. Cree un rol de IAM y asócielo a una cuenta de servicio de Kubernetes para el controlador de Spark

```
eksctl create iamserviceaccount --name my-spark-driver-sa --namespace spark-operator \
--cluster my-cluster --role-name "my-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

4. Cree un archivo YAML con los [permisos](#) necesarios para la cuenta de servicio del controlador de Spark:

```
cat >spark-rbac.yaml <<EOF
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: emr-containers-role-spark
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-role-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: emr-containers-role-spark
subjects:
- kind: ServiceAccount
  name: emr-containers-sa-spark
  namespace: default
```

```
EOF
```

5. Aplique las configuraciones de la vinculación de roles del clúster.

```
kubectl apply -f spark-rbac.yaml
```

6. El comando `kubectl` debería devolver la confirmación de la cuenta creada.

```
serviceaccount/emr-containers-sa-spark created  
clusterrolebinding.rbac.authorization.k8s.io/emr-containers-role-spark configured
```

Ejecución de la aplicación de Spark

Amazon EMR 6.10.0 y las versiones posteriores admiten `spark-submit` para ejecutar aplicaciones de Spark en un clúster de Amazon EKS. Complete los pasos que se indican a continuación para ejecutar la aplicación de Spark:

1. Asegúrese de que ha completado los pasos de [Configuración de spark-submit para Amazon EMR en EKS](#).
2. Establezca los valores de las siguientes variables de entorno:

```
export SPARK_HOME=spark-home  
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

3. Luego, envíe la solicitud de Spark con el siguiente comando:

```
$SPARK_HOME/bin/spark-submit \  
  --class org.apache.spark.examples.SparkPi \  
  --master $MASTER_URL \  
  --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-  
west-2.amazonaws.com/spark/emr-6.15.0:latest \  
  --conf spark.kubernetes.authenticate.driver.serviceAccountName=emr-containers-sa-  
spark \  
  --deploy-mode cluster \  
  --conf spark.kubernetes.namespace=default \  
  --conf "spark.driver.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/  
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/  
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/  
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-  
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
```

```

serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
  --conf "spark.driver.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native" \
  --conf "spark.executor.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
  --conf "spark.executor.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/
hadoop-lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/
lib/native" \
  --conf
spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.auth.WebIdentityTokenCredent
\
  --conf spark.hadoop.fs.s3.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem \
  --conf
spark.hadoop.fs.AbstractFileSystem.s3.impl=org.apache.hadoop.fs.s3.EMRFSDelegate \
  --conf spark.hadoop.fs.s3.buffer.dir=/mnt/s3 \
  --conf spark.hadoop.fs.s3.getObject.initialSocketTimeoutMilliseconds="2000" \
  --conf
spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFile
\
  --conf spark.hadoop.mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem="true" \
  s3://my-pod-bucket/spark-examples.jar 20

```

4. Cuando el controlador de Spark finalice el trabajo de Spark, debería ver una línea de registro al final del envío que indica que el trabajo de Spark ha finalizado.

```

23/11/24 17:02:14 INFO LoggingPodStatusWatcherImpl: Application
org.apache.spark.examples.SparkPi with submission ID default:org-apache-spark-
examples-sparkpi-4980808c03ff3115-driver finished
23/11/24 17:02:14 INFO ShutdownHookManager: Shutdown hook called

```

Limpieza

Cuando termine de ejecutar las aplicaciones, puede realizar la limpieza con el siguiente comando.


```
kubectl delete -f spark-rbac.yaml
```

Uso de Apache Livy con Amazon EMR en EKS

Con las versiones 7.1.0 y posteriores de Amazon EMR, puede utilizar Apache Livy para enviar trabajos en Amazon EMR en EKS. Mediante Apache Livy, puede configurar su propio punto de conexión REST de Apache Livy y usarlo para implementar y administrar aplicaciones de Spark en sus clústeres de Amazon EKS. Una vez que haya instalado Livy en su clúster de Amazon EKS, puede utilizar el punto de conexión de Livy para enviar las aplicaciones de Spark a su servidor de Livy. El servidor administra el ciclo de vida de las aplicaciones de Spark.

Note

Amazon EMR calcula los precios de Amazon EKS en función de la vCPU y el consumo de memoria. Este cálculo se aplica a los pods de controladores y ejecutores. Este cálculo comienza cuando usted descarga la imagen de la aplicación de Amazon EMR hasta que finaliza el pod de Amazon EKS y se redondea al segundo más cercano.

Temas

- [Configuración de Apache Livy para Amazon EMR en EKS](#)
- [Introducción a Apache Livy en Amazon EMR en EKS](#)
- [Ejecución de una aplicación de Spark con Apache Livy para Amazon EMR en EKS](#)
- [Desinstalación de Apache Livy con Amazon EMR en EKS](#)
- [Seguridad de Apache Livy con Amazon EMR en EKS](#)
- [Propiedades de instalación de Apache Livy en las versiones de Amazon EMR en EKS.](#)
- [Solucione los errores comunes de formato de las variables de entorno](#)

Configuración de Apache Livy para Amazon EMR en EKS

Antes de poder instalar Apache Livy en el clúster de Amazon EKS, debe instalar y configurar un conjunto de herramientas con requisitos previos. Estas incluyen la AWS CLI, que es una herramienta de línea de comandos básica para trabajar con AWS recursos de, herramientas de línea de comandos para trabajar con Amazon EKS y un controlador que se utiliza en este caso de uso para hacer que la aplicación de clúster esté disponible en Internet y enrutar el tráfico de la red.

- [Instale o actualice a la versión más reciente de AWS CLI: si ya la](#) ha instalado AWS CLI, confirme que tiene la última versión.
- [Configure kubectl y eksctl](#) : eksctl es una herramienta de línea de comandos que se utiliza para comunicarse con Amazon EKS.
- [Instale Helm](#): el administrador de paquetes Helm para Kubernetes le ayuda a instalar y administrar aplicaciones en el clúster de Kubernetes.
- [Introducción a Amazon EKS: eksctl](#) : siga los pasos para crear un nuevo clúster de Kubernetes con nodos en Amazon EKS.
- [Seleccione una etiqueta de versión de Amazon EMR](#): el Apache Livy es compatible con las versiones 7.1.0 y superiores de Amazon EMR.
- [Instale el controlador ALB: el controlador](#) ALB administra AWS Elastic Load Balancing de para los clústeres de Kubernetes. Crea un Equilibrador de carga de AWS red (NLB) de al crear un Equilibrador de carga de red (NLB) de al crear un Equilibrador de carga de red (NLB) de al crear un Kubernetes Livy.

Introducción a Apache Livy en Amazon EMR en EKS

Siga los pasos que se describen a continuación para instalar Apache Livy. Incluyen la configuración del administrador de paquetes, la creación de un espacio de nombres para ejecutar las cargas de trabajo de Spark, la instalación de Livy, la configuración del equilibrio de carga y los pasos de verificación. Tiene que completar estos pasos para ejecutar un trabajo por lotes con Spark.

1. Si aún no lo ha hecho, configure [Apache Livy para Amazon EMR en EKS](#).
2. Autentique su cliente de Helm en el registro de Amazon ECR. Puede encontrar el ECR-registry-account valor ECR-registry-account correspondiente Región de AWS a su a partir de sus [cuentas de registro de Amazon ECR por región](#).

```
aws ecr get-login-password --region <AWS_REGION> | helm registry login \
--username AWS \
--password-stdin <ECR-registry-account>.dkr.ecr.<region-id>.amazonaws.com
```

3. Al configurar Livy, se crea una cuenta de servicio para el servidor de Livy y otra cuenta para la aplicación Spark. Para configurar IRSA para las cuentas de servicio, consulte [Configuración de los permisos de acceso con roles de IAM para las cuentas de servicio \(IRSA\)](#).
4. Cree un espacio de nombres para ejecutar sus cargas de trabajo de Spark.

```
kubectl create ns <spark-ns>
```

5. Utilice el siguiente comando para instalar Livy.

Este punto de conexión de Livy solo está disponible internamente para la VPC del clúster de EKS. Para habilitar el acceso más allá de la VPC, establezca `--set loadbalancer.internal=false` en su comando de instalación de Helm.

 Note

De forma predeterminada, el SSL no está habilitado en este punto de conexión de Livy y el punto de conexión solo está visible dentro de la VPC del clúster de EKS. Si configura `loadbalancer.internal=false` y `ssl.enabled=false`, expone un punto de conexión inseguro al exterior de su VPC. Para configurar un punto de conexión Livy seguro, consulte [Configuración de un punto de conexión seguro de Apache Livy con TLS/SSL](#).

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.9.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.9.0:latest \
  --set sparkNamespace=<spark-ns> \
  --create-namespace
```

Debería ver la siguiente salida.

```
NAME: livy-demo
LAST DEPLOYED: Mon Mar 18 09:23:23 2024
NAMESPACE: livy-ns
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Livy server has been installed.
Check installation status:
1. Check Livy Server pod is running
```

```
kubectl --namespace livy-ns get pods -l "app.kubernetes.io/instance=livy-demo"
```

2. Verify created NLB is in Active state and it's target groups are healthy (if loadbalancer.enabled is true)

Access LIVY APIs:

```
# Ensure your NLB is active and healthy
# Get the Livy endpoint using command:
LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/instance=livy-demo,emr-containers.amazonaws.com/type=loadbalancer -o jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' | awk '{printf "%s:8998\n", $0}')
# Access Livy APIs using http://$LIVY_ENDPOINT or https://$LIVY_ENDPOINT (if SSL is enabled)
# Note: While uninstalling Livy, makes sure the ingress and NLB are deleted after running the helm command to avoid dangling resources
```

Los nombres de las cuentas de servicio predeterminados para el servidor de Livy y la sesión de Spark son `emr-containers-sa-livy` y `emr-containers-sa-spark-livy`. Para usar nombres personalizados, use los parámetros `serviceAccounts.name` y `sparkServiceAccount.name`.

```
--set serviceAccounts.name=my-service-account-for-livy
--set sparkServiceAccount.name=my-service-account-for-spark
```

6. Compruebe que haya instalado el gráfico de Helm.

```
helm list -n livy-ns -o yaml
```

El comando `helm list` debería devolver la información sobre el nuevo gráfico de Helm:

```
app_version: 0.7.1-incubating
chart: livy-emr-7.9.0
name: livy-demo
namespace: livy-ns
revision: "1"
status: deployed
updated: 2024-02-08 22:39:53.539243 -0800 PST
```

7. Compruebe que el Equilibrador de carga de red esté activo.

```
LIVY_NAMESPACE=<livy-ns>
```

```
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB Endpoint URL
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the status of the NLB that matching the endpoint from the Kubernetes service
NLB_STATUS=$(echo $ELB_LIST | grep -A 8 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk '/Code/{print $2}/' | tr -d '"',\n')
echo $NLB_STATUS
```

8. Ahora compruebe que el grupo objetivo del Equilibrador de carga de red esté en buen estado.

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB endpoint
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the NLB ARN from the NLB endpoint
NLB_ARN=$(echo $ELB_LIST | grep -B 1 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk '/"LoadBalancerArn":/,/,/' | awk '/:/{print $2}' | tr -d '\",)')

# Get the target group from the NLB. Livy setup only deploys 1 target group
TARGET_GROUP_ARN=$(aws elbv2 describe-target-groups --load-balancer-arn $NLB_ARN --region $AWS_REGION | awk '/"TargetGroupArn":/,/,/' | awk '/:/{print $2}' | tr -d '\",)')

# Get health of target group
aws elbv2 describe-target-health --target-group-arn $TARGET_GROUP_ARN
```

Aquí se incluye un resultado de ejemplo que muestra el estado del grupo objetivo:

```
{
  "TargetHealthDescriptions": [
    {
      "Target": {
        "Id": "<target IP>",
        "Port": 8998,
        "AvailabilityZone": "us-west-2d"
      },
      "HealthCheckPort": "8998",
      "TargetHealth": {
        "State": "healthy"
      }
    }
  ]
}
```

Una vez que el estado de su NLB pase a estar activo y su grupo objetivo esté healthy, podrá continuar. Puede que tarde unos minutos.

9. Recupere el punto de conexión de Livy de la instalación de Helm. El hecho de que su punto de conexión de Livy sea seguro o no depende de si ha habilitado el SSL.

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=livy-app-name
LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-app-name,emr-containers.amazonaws.com/type=loadbalancer -o
jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' | awk '{printf
"%s:8998\n", $0}')
echo "$LIVY_ENDPOINT"
```

10. Recupere la cuenta de servicio de Spark de la instalación de Helm

```
SPARK_NAMESPACE=spark-ns
LIVY_APP_NAME=<livy-app-name>
SPARK_SERVICE_ACCOUNT=$(kubectl --namespace $SPARK_NAMESPACE
get sa -l "app.kubernetes.io/instance=$LIVY_APP_NAME" -o
jsonpath='{.items[0].metadata.name}')
echo "$SPARK_SERVICE_ACCOUNT"
```

Debería ver algo similar al siguiente resultado:

`emr-containers-sa-spark-livy`

11. Si ha establecido `internalALB=true` habilitar el acceso desde fuera de la VPC, cree una EC2 instancia de Amazon EC2 y asegúrese de que el Equilibrador de carga de red permita el tráfico de red procedente de la instancia de EC2. EC2 Debe hacerlo para que la instancia pueda acceder a su punto de conexión de Livy. Para obtener más información sobre cómo exponer su punto de conexión de forma segura fuera de su VPC, consulte [Configuración con un punto de conexión de Apache Livy seguro con TLS/SSL](#).
12. Al instalar Livy, se crea la cuenta de servicio `emr-containers-sa-spark` para ejecutar las aplicaciones de Spark. Si su aplicación de Spark cualquier AWS recurso de como S3 o llama a operaciones de la AWS API o la CLI de, debe vincular un rol de IAM con los permisos necesarios a su cuenta de servicio de Spark. Para obtener más información, consulte [Configuración de los permisos de acceso con roles de IAM para las cuentas de servicio \(IRSA\)](#).

Apache Livy admite configuraciones adicionales que puede usar al instalar Livy. Para obtener más información, consulte Propiedades de instalación de Apache Livy en Amazon EMR en EKS.

Ejecución de una aplicación de Spark con Apache Livy para Amazon EMR en EKS

Antes de poder ejecutar una aplicación de Spark con Apache Livy, asegúrese de haber completado los pasos de [Configuración de Apache Livy para Amazon EMR en EKS](#) y la [Introducción a Apache Livy para Amazon EMR en EKS](#).

Puede usar Apache Livy para ejecutar dos tipos de aplicaciones:

- Sesiones por lotes: un tipo de carga de trabajo de Livy para enviar trabajos por lotes a Spark.
- Sesiones interactivas: un tipo de carga de trabajo de Livy que proporciona una interfaz visual y programática para ejecutar consultas de Spark.

Note

Los pods controladores y ejecutores de sesiones diferentes pueden comunicarse entre sí. Los espacios de nombres no garantizan la seguridad entre los pods. Kubernetes no

permite permisos selectivos en un subconjunto de pods dentro de un espacio de nombres determinado.

Ejecución de sesiones por lotes

Para enviar un trabajo por lotes, use el siguiente comando.

```
curl -s -k -H 'Content-Type: application/json' -X POST \
  -d '{
    "name": "my-session",
    "file": "entryPoint_location (S3 or local)",
    "args": ["argument1", "argument2", ...],
    "conf": {
      "spark.kubernetes.namespace": "<spark-namespace>",
      "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.9.0:latest",
      "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-
service-account>"
    }
  }' <livy-endpoint>/batches
```

Para monitorizar el trabajo por lotes, use el siguiente comando.

```
curl -s -k -H 'Content-Type: application/json' -X GET <livy-endpoint>/batches/my-
session
```

Ejecución de sesiones interactivas

Para ejecutar sesiones interactivas con Apache Livy, consulte los siguientes pasos.

1. Asegúrese de tener acceso a un cuaderno de Jupyter de Jupyter, como un cuaderno de Jupyter de Jupyter de Spark SageMaker . Su cuaderno de Jupyter debe tener [sparkmagic](#) instalado.
2. Cree un bucket para la configuración de Spark `spark.kubernetes.file.upload.path`. Asegúrese de que la cuenta de servicio de Spark tenga acceso de lectura y escritura al bucket. Para obtener más información sobre cómo configurar su cuenta de servicio de Spark, consulte Configuración de los permisos de acceso con roles de IAM para las cuentas de servicio (IRSA)
3. Cargue `sparkmagic` en el cuaderno de Jupyter con el comando `%load_ext sparkmagic.magics`.

4. Ejecute el comando `%manage_spark` para configurar su punto de conexión de Livy con el cuaderno de Jupyter. Seleccione la pestaña Añadir puntos de conexión, elija el tipo de autenticación configurado, añada el punto de conexión de Livy al cuaderno y, a continuación, elija Añadir punto de conexión.
5. Vuelva a ejecutar `%manage_spark` para crear el contexto de Spark y, a continuación, vaya a Crear sesión. Elija el punto de conexión de Livy, especifique un nombre de sesión único, elija un idioma y, a continuación, añada las siguientes propiedades.

```
{
  "conf": {
    "spark.kubernetes.namespace": "livy-namespace",
    "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.9.0:latest",
    "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-service-
account>",
    "spark.kubernetes.file.upload.path": "<URI_TO_S3_LOCATION>"
  }
}
```

6. Envíe la solicitud y espere a que cree el contexto de Spark.
7. Para monitorizar el estado de la sesión interactiva, ejecute el siguiente comando.

```
curl -s -k -H 'Content-Type: application/json' -X GET livy-endpoint/sessions/my-
interactive-session
```

Monitorización de aplicaciones de Spark

Para monitorizar el progreso de sus aplicaciones de Spark con la IU de Livy, use el enlace `http://<livy-endpoint>/ui`.

Desinstalación de Apache Livy con Amazon EMR en EKS

Siga los pasos a continuación para desinstalar Apache Livy.

1. Elimine la configuración de Livy con los nombres de su espacio de nombres y el nombre de la aplicación. En este ejemplo, el nombre de la aplicación es `livy-demo` y el espacio de nombres es `livy-ns`.

```
helm uninstall livy-demo -n livy-ns
```

2. Al realizar la desinstalación, Amazon EMR en EKS elimina el servicio de Kubernetes en Livy, los equilibradores de carga de y AWS los grupos objetivo que creó durante la instalación. La eliminación de recursos puede tardar unos minutos. Asegúrese de que los recursos se eliminen antes de volver a instalar Livy en el espacio de nombres.
3. Elimina el espacio de nombres de Spark.

```
kubectl delete namespace spark-ns
```

Seguridad de Apache Livy con Amazon EMR en EKS

Consulte los siguientes temas para obtener más información sobre la configuración de la seguridad de Apache Livy con Amazon EMR en EKS. Estas opciones incluyen el uso de la seguridad a nivel de capa de transporte, el control de acceso basado en roles, es decir, el acceso en función del rol de una persona dentro de una organización, y el uso de roles de IAM, que proporcionan acceso a los recursos en función de los permisos otorgados.

Temas

- [Configuración de un punto de conexión de Apache Livy seguro con TLS/SSL](#)
- [Configuración de los permisos de las aplicaciones de Apache Livy y Spark con control de acceso basado en roles \(RBAC\)](#)
- [Configuración de los permisos de acceso con roles de IAM para las cuentas de servicio \(IRSA\)](#)

Configuración de un punto de conexión de Apache Livy seguro con TLS/SSL

Consulte las siguientes secciones para obtener más información sobre la configuración de Livy para Amazon EMR en EKS end-to-end con cifrado TLS y SSL de extremo a extremo.

Configuración del cifrado TLS y SSL

Para configurar el cifrado SSL en su punto de conexión de Apache Livy, siga estos pasos.

- [Instale el controlador CSI de Secrets Store y el Secrets Store y el AWS Secrets Store y el Secrets Store y el Secrets Store y el ASKP](#) almacenan de forma segura los certificados

JKS y las contraseñas de Livy que el pod del servidor de Livy necesita para habilitar el SSL.

También puede instalar solo el controlador CSI de Secrets Store y utilizar cualquier otro proveedor de secretos compatible.

- [Cree un certificado ACM](#): este certificado es necesario para proteger la conexión entre el cliente y el punto de conexión de ALB.
- Configure un certificado JKS, una contraseña clave y una contraseña del almacén de claves para AWS Secrets Manager : necesarios para proteger la conexión entre el punto de conexión de ALB y el servidor de Livy.
- Añada permisos a la cuenta de servicio de Livy para recuperar los secretos de AWS Secrets Manager : el servidor de Livy necesita estos permisos para recuperar los secretos del ASCP y añadir las configuraciones de Livy para proteger el servidor de Livy. Para añadir permisos de IAM a la cuenta del servicio, consulte Configuración de los permisos de acceso con roles de IAM para las cuentas de servicio (IRSA).

Configuración de un certificado JKS con una clave y una contraseña de almacén de claves para AWS Secrets Manager

Siga estos pasos para configurar un certificado JKS con una clave y una contraseña de almacén de claves.

1. Genere un archivo de almacén de claves para el servidor de Livy.

```
keytool -genkey -alias <host> -keyalg RSA -keysize 2048 -dname  
CN=<host>,OU=hw,O=hw,L=<your_location>,ST=<state>,C=<country> -  
keypass <keyPassword> -keystore <keystore_file> -storepass <storePassword> --  
validity 3650
```

2. Cree un certificado.

```
keytool -export -alias <host> -keystore mykeystore.jks -rfc -  
file mycertificate.cert -storepass <storePassword>
```

3. Cree un archivo de almacén de confianza.

```
keytool -import -noprompt -alias <host>-file <cert_file> -  
keystore <truststore_file> -storepass <truststorePassword>
```

4. Guarde el certificado JKS en AWS Secrets Manager. Sustituya `livy-jks-secret` por su secreto y `fileb://mykeystore.jks` por la ruta al certificado JKS de su almacén de claves.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy keystore JKS secret" \
--secret-binary fileb://mykeystore.jks
```

5. Guarde el almacén de claves y la contraseña de claves en Secrets Manager. Asegúrese de utilizar sus propios parámetros.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy key and keystore password secret" \
--secret-string "{\"keyPassword\": \"<test-key-password>\", \"keyStorePassword\": \"<test-key-store-password>\"}"
```

6. Cree un espacio de nombres del servidor de Livy con el siguiente comando.

```
kubectl create ns <livy-ns>
```

7. Cree el objeto ServiceProviderClass para el servidor de Livy que tiene el certificado JKS y las contraseñas.

```
cat >livy-secret-provider-class.yaml << EOF
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "livy-jks-secret"
        objectType: "secretsmanager"
      - objectName: "livy-passwords"
        objectType: "secretsmanager"

EOF
kubectl apply -f livy-secret-provider-class.yaml -n <livy-ns>
```

Introducción a Apache Livy con SSL habilitado

Después de habilitar el SSL en su servidor de Livy, debe configurar la `serviceAccount` para tener acceso a los secretos `keyStore` y `keyPasswords` en AWS Secrets Manager.

1. Cree el espacio de nombres del servidor de Livy.

```
kubectl create namespace <livy-ns>
```

2. Configure la cuenta de servicio Livy para tener acceso a los secretos en Secrets Manager. Para obtener más información sobre la configuración de IRSA, consulte [Configuración de los IRSA durante la instalación de Apache Livy](#).

```
aws ecr get-login-password --region region-id | helm registry login \
--username AWS \
--password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
```

3. Instale Livy. Para el gráfico de Helm: parámetro de versión, utilice su etiqueta de versión de Amazon EMR, como 7.1.0. También debe sustituir el ID de cuenta del registro de Amazon ECR y el ID de región por sus propios IDs ID. Puede encontrar el ECR-registry-account valor ECR-registry-account correspondiente Región de AWS a su a partir de sus [cuentas de registro de Amazon ECR por región](#).

```
helm install <livy-app-name> \
oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
--version 7.9.0 \
--namespace livy-namespace-name \
--set image=<ECR-registry-account.dkr.ecr>.<region>.amazonaws.com/livy/
emr-7.9.0:latest \
--set sparkNamespace=spark-namespace \
--set ssl.enabled=true
--set ssl.CertificateArn=livy-acm-certificate-arn
--set ssl.secretProviderClassName=aws-secrets
--set ssl.keyStoreObjectName=livy-jks-secret
--set ssl.keyPasswordsObjectName=livy-passwords
--create-namespace
```

4. Continúe con el paso 5 de la [Instalación de Apache Livy en Amazon EMR en EKS](#).

Configuración de los permisos de las aplicaciones de Apache Livy y Spark con control de acceso basado en roles (RBAC)

Para implementar Livy, Amazon EMR en EKS crea una cuenta y un rol de servicio de servidor y un rol y una cuenta de servicio de Spark. Estos roles deben tener los permisos RBAC necesarios para finalizar la configuración y ejecutar las aplicaciones de Spark.

Permisos RBAC para el rol y la cuenta de servicio del servidor

Amazon EMR en EKS crea el rol y la cuenta de servicio de servidor de Livy para administrar sesiones de Livy para los trabajos de Spark y enrutar el tráfico hacia y desde los recursos de entrada y otros recursos.

El nombre predeterminado de esta cuenta de servicio es `emr-containers-sa-livy`. Debe tener los siguientes permisos.

```
rules:
- apiGroups:
  - ""
  resources:
  - "namespaces"
  verbs:
  - "get"
- apiGroups:
  - ""
  resources:
  - "serviceaccounts"
  - "services"
  - "configmaps"
  - "events"
  - "pods"
  - "pods/log"
  verbs:
  - "get"
  - "list"
  - "watch"
  - "describe"
  - "create"
  - "edit"
  - "delete"
  - "deletecollection"
  - "annotate"
```

```
    "patch"
    "label"
  - apiGroups:
    - ""
    resources:
    - "secrets"
    verbs:
    - "create"
    "patch"
    "delete"
    "watch"
  - apiGroups:
    - ""
    resources:
    - "persistentvolumeclaims"
    verbs:
    - "get"
    "list"
    "watch"
    "describe"
    "create"
    "edit"
    "delete"
    "annotate"
    "patch"
    "label"
```

Permisos RBAC para el rol y la cuenta de servicio de Spark

Un pod controlador de Spark necesita una cuenta de servicio de Kubernetes en el mismo espacio de nombres que el pod. Esta cuenta de servicio necesita permisos para administrar los pods ejecutores y cualquier recurso que requiera el pod controlador. A menos que la cuenta de servicio predeterminada del espacio de nombres tenga los permisos necesarios, el controlador fallará y se cerrará. Los siguientes permisos RBAC son necesarios.

```
rules:
- apiGroups:
  - ""
  "batch"
  "extensions"
  "apps"
  resources:
  - "configmaps"
```

```
"serviceaccounts"  
"events"  
"pods"  
"pods/exec"  
"pods/log"  
"pods/portforward"  
"secrets"  
"services"  
"persistentvolumeclaims"  
"statefulsets"  
verbs:  
- "create"  
  "delete"  
  "get"  
  "list"  
  "patch"  
  "update"  
  "watch"  
  "describe"  
  "edit"  
  "deletecollection"  
  "patch"  
  "label"
```

Configuración de los permisos de acceso con roles de IAM para las cuentas de servicio (IRSA)

De forma predeterminada, el servidor de Livy y el controlador y ejecutores de la aplicación Spark no tienen acceso a AWS los recursos de. La cuenta de servicio del servidor y la cuenta de servicio Spark controlan el acceso a AWS los recursos de de los pods del servidor de Livy y de la aplicación Spark. Para conceder el acceso, debe asignar las cuentas de servicio a un rol de IAM que tenga los AWS permisos necesarios de.

Puede configurar la asignación de IRSA antes de instalar Apache Livy, durante la instalación o después de finalizarla.

Configuración de IRSA durante la instalación de Apache Livy (para la cuenta de servicio del servidor)

Note

Esta asignación solo es compatible con la cuenta de servicio del servidor.

1. Asegúrese de haber terminado la [configuración de Apache Livy para Amazon EMR en EKS](#) y de que está realizando la [instalación de Apache Livy con Amazon EMR en EKS](#).
2. Cree un espacio de nombres de Kubernetes para el servidor Livy. En este ejemplo, el nombre del espacio de nombres es `livy-ns`.
3. Cree una política de IAM que incluya los permisos Servicios de AWS para los a los que quiere que accedan sus pods. El siguiente ejemplo crea una política de IAM para obtener recursos de Amazon S3 para el punto de entrada de Spark.

```
cat >my-policy.json <<EOF{
  "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
          "Action": "s3:GetObject",
          "Resource": "arn:aws:s3:::my-spark-entriypoint-bucket"
        }
      ]
    }
  EOF

aws iam create-policy --policy-name my-policy --policy-document file:///my-
policy.json
```

4. Use el siguiente comando para establecer su Cuenta de AWS ID de en una variable.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

5. Establezca el proveedor de identidades de OpenID Connect (OIDC) de su clúster en una variable de entorno.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\\\\//")
```

6. Establezca variables para el espacio de nombres y el nombre de la cuenta de servicio. Asegúrese de utilizar sus propios valores.

```
export namespace=default
export service_account=my-service-account
```

7. Cree un archivo de política de confianza con el comando siguiente. Si desea conceder el acceso al rol a todas las cuentas de servicio de un espacio de nombres, copie el siguiente comando y sustituya `StringEquals` por `StringLike`; sustituya, también, `$service_account` por `*`.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

8. Creación del rol.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-relationship.json --description "my-role-description"
```

9. Utilice el siguiente comando de instalación Helm para establecer el `serviceAccount.executionRoleArn` para asignar IRSA. A continuación, se muestra un ejemplo del comando de instalación Helm. Puede encontrar el ECR-registry-account valor ECR-registry-account correspondiente Región de AWS a su a partir de sus [cuentas de registro de Amazon ECR por región](#).

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.us-west-2.amazonaws.com/livy \
  --version 7.9.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
  emr-7.9.0:latest \
```

```
--set sparkNamespace=spark-ns \  
--set serviceAccount.executionRoleArn=arn:aws:iam::123456789012:role/my-role
```

Asignación del IRSA a una cuenta de servicio de Spark

Antes de asignar el IRSA a una cuenta de servicio de Spark, asegúrese de haber realizado los siguientes pasos:

- Asegúrese de haber terminado la [configuración de Apache Livy para Amazon EMR en EKS](#) y de que está realizando la [instalación de Apache Livy con Amazon EMR en EKS](#).
- Debe tener un proveedor de OpenID Connect (OIDC) de IAM para su clúster. Para ver si ya tiene este proveedor o cómo crear uno, consulte [Crear un proveedor de OIDC de IAM para su clúster](#).
- Asegúrese de que tiene la versión 0.171.0 o posterior de la CLI de eksctl instalada o AWS CloudShell. Para instalar o actualizar eksctl, consulte [Instalación](#) en la eksctl documentación.

Siga estos pasos para asignar el IRSA a su cuenta de servicio de Spark:

1. Ejecute el comando siguiente para obtener la cuenta de servicio de Spark.

```
SPARK_NAMESPACE=<spark-ns>  
LIVY_APP_NAME=<livy-app-name>  
kubectl --namespace $SPARK_NAMESPACE describe sa -l "app.kubernetes.io/instance=  
$LIVY_APP_NAME" | awk '/^Name:/ {print $2}'
```

2. Establezca variables para el espacio de nombres y el nombre de la cuenta de servicio.

```
export namespace=default  
export service_account=my-service-account
```

3. Use el siguiente comando para crear un archivo de política de confianza para el rol de IAM. El siguiente ejemplo da permiso a todas las cuentas de servicio dentro del espacio de nombres para utilizar el rol. Para ello, sustituya StringEquals por StringLike; sustituya, también, \$service_account por *.

```
cat >trust-relationship.json <<EOF  
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  

```

```

    "Effect": "Allow",
    "Principal": {
      "Federated": "arn:aws:iam::${account_id}:oidc-provider/${oidc_provider}"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "${oidc_provider}:aud": "sts.amazonaws.com",
        "${oidc_provider}:sub": "system:serviceaccount:${namespace}:${service_account}"
      }
    }
  }
]
}
EOF

```

4. Creación del rol.

```

aws iam create-role --role-name my-role --assume-role-policy-document file://trust-relationship.json --description "my-role-description"

```

5. Asigne el servidor o la cuenta de servicio de Spark con el siguiente comando eksctl. Asegúrese de utilizar sus propios valores.

```

eksctl create iamserviceaccount --name spark-sa \
--namespace spark-namespace --cluster livy-eks-cluster \
--attach-role-arn arn:aws:iam::0123456789012:role/my-role \
--approve --override-existing-serviceaccounts

```

Propiedades de instalación de Apache Livy en las versiones de Amazon EMR en EKS.

La instalación de Apache Livy le permite seleccionar una versión del gráfico de Helm de Livy. El gráfico de Helm ofrece una variedad de propiedades para personalizar su experiencia de instalación y configuración. Estas propiedades son compatibles con las versiones 7.1.0 y posteriores de Amazon EMR en EKS.

Temas

- [Propiedades de instalación de Amazon EMR 7.1.0](#)

Propiedades de instalación de Amazon EMR 7.1.0

La tabla siguiente describe todas las propiedades de Livy compatibles. Al instalar Apache Livy, puede elegir la versión del gráfico de Helm de Livy. Para establecer una propiedad durante la instalación, utilice el comando `--set <property>=<value>`.

Propiedad	Descripción	Predeterminado
imagen	El URI de la versión de Amazon EMR del servidor Livy. Se trata de una configuración obligatoria.	""
sparkNamespace	Espacio de nombres para ejecutar sesiones de Spark de Livy. Por ejemplo, especifique "livy". Se trata de una configuración obligatoria.	""
nameOverride	Proporcione un nombre en lugar de livy. El nombre se establece como una etiqueta para todos los recursos de Livy	"livy"
fullnameOverride	Proporcione un nombre para usarlo en lugar de los nombres completos de los recursos.	""
ssl.enabled	Habilita el end-to-end SSL de extremo a extremo desde el punto de conexión de Livy al servidor Livy.	FALSO
ssl.certificateArn	Si el SSL está habilitado, este es el ARN del certificado ACM	""

Propiedad	Descripción	Predeterminado
	para el NLB creado por el servicio.	
ssl secretProviderClassNombre	Si el SSL está habilitado, este es el nombre de clase del proveedor secreto para proteger el NLB para la conexión del servidor Livy con SSL.	""
ssl keyStoreObjectName	Si el SSL está habilitado, este es el nombre del objeto del certificado del almacén de claves de la clase de proveedor secreto.	""
ssl keyPasswordsObjectName	Si el SSL está activado, este es el nombre de objeto del secreto que contiene el almacén de claves y la contraseña de claves.	""
rbac.create	Si es verdadero, crea recursos RBAC.	FALSO
serviceAccount.create	Si es verdadero, crea una cuenta de servicio de Livy.	TRUE

Propiedad	Descripción	Predeterminado
<code>serviceAccount.name</code>	El nombre de la cuenta de servicio que se utilizará para Livy. Si no establece esta propiedad ni crea una cuenta de servicio, Amazon EMR en EKS generará automáticamente un nombre mediante la propiedad de anulación de <code>fullname</code> .	"emr-containers-sa-livy"
Cuenta de servicio. <code>executionRoleArn</code>	El ARN del rol de ejecución de la cuenta de servicio de Livy.	""
<code>sparkServiceAccount.crear</code>	Si es verdadero, crea la cuenta de servicio de Spark en <code>.Release.Namespace</code>	TRUE
<code>sparkServiceAccount.name</code>	El nombre de la cuenta de servicio que se utilizará para Spark. Si no establece esta propiedad ni crea una cuenta de servicio Spark, Amazon EMR en EKS generará automáticamente un nombre mediante la propiedad <code>fullnameOverride</code> con el sufijo <code>-spark-livy</code> .	«emr-containers-sa-spark-livy»
<code>service.name</code>	Nombre del servicio de Livy	"emr-containers-livy"
<code>service.annotations</code>	Anotaciones del servicio de Livy	{}

Propiedad	Descripción	Predeterminado
loadbalancer.enabled	Si se debe crear un equilibrador de carga para el servicio Livy utilizado para exponer el punto de conexión de Livy fuera del clúster de Amazon EKS.	FALSE
loadbalancer.internal	Si se debe configurar el punto de conexión de Livy como interno en la VPC o externo. Si se establece esta propiedad en FALSE se expone el punto de conexión a fuentes externas a la VPC. Recomendamos proteger el punto de conexión con TLS/SSL. Para obtener más información, consulte Configuración del cifrado TLS y SSL.	FALSE
imagePullSecrets	La lista de nombres <code>imagePullSecret</code> que se utilizarán para extraer la imagen de Livy de los repositorios privados.	[]
recursos	Las solicitudes de recursos y los límites de los contenedores de Livy.	{}
nodeSelector	Los nodos para los que programar los pods de Livy.	{}

Propiedad	Descripción	Predeterminado
toleraciones	Una lista que contiene las tolerancias de los pods de Livy a definir.	[]
afinidad	Las reglas de afinidad de los pods de Livy.	{}
persistence.enabled	Si es verdadero, habilita la persistencia de los directorios de sesiones.	FALSO
persistence.subPath	La subruta de PVC que se monta en los directorios de las sesiones.	""
persistence.existingClaim	El PVC a utilizar en lugar de crear uno nuevo.	{}
persistence.storageClass	La clase de almacenamiento a utilizar. Para definir este parámetro, utilice el formato <code>storageClassName: <storageClass></code> . Si este parámetro se establece en "- ", se deshabilita el aprovisionamiento dinámico. Si establece este parámetro en «null» o no especifica nada, Amazon EMR en EKS no establece un <code>storageClassName</code> Equilibrador predeterminado.	""
persistence.accessMode	El modo de acceso al PVC.	ReadWriteOnce
persistence.size	El tamaño del PVC.	20 Gi

Propiedad	Descripción	Predeterminado
<code>persistence.annotations</code>	Anotaciones adicionales para el PVC.	<code>{}</code>
<code>env.*</code>	Envs adicionales para establecer en el contenedor de Livy. Para obtener más información, consulte Introducción de sus propias configuraciones de Livy y Spark durante la instalación de Livy .	<code>{}</code>
<code>envFrom.*</code>	Envs adicionales para establecer Livy desde un mapa de configuración o secreto.	<code>[]</code>
<code>livyConf.*</code>	Entradas <code>livy.conf</code> adicionales para establecer desde un mapa de configuración o secreto de Kubernetes montado.	<code>{}</code>
<code>sparkDefaultsConf.*</code>	Entradas <code>spark-defaults.conf</code> adicionales para establecer desde un mapa de configuración o secreto de Kubernetes montado.	<code>{}</code>

Solucione los errores comunes de formato de las variables de entorno

Al introducir las configuraciones de Livy y Spark, hay formatos de variables de entorno que no son compatibles y pueden provocar errores. El procedimiento le guía por una serie de pasos que le ayudan a garantizar que utiliza los formatos correctos.

Introducción de sus propias configuraciones de Livy y Spark durante la instalación de Livy

Puede configurar cualquier variable de entorno de Apache Livy o Apache Spark con la propiedad Helm `env.*`. Siga los pasos que se indican a continuación para convertir la configuración de ejemplo `example.config.with-dash.withUppercase` a un formato de variable de entorno compatible.

1. Sustituya las letras mayúsculas por un 1 y la letra en minúscula. Por ejemplo, `example.config.with-dash.withUppercase` se convierte en `example.config.with-dash.with1uppercase`.
2. Sustituya los guiones (-) por 0. Por ejemplo, `example.config.with-dash.with1uppercase` se convierte en `example.config.with0dash.with1uppercase`.
3. Sustituya los puntos (.) por guiones bajos (_). Por ejemplo, `example.config.with0dash.with1uppercase` se convierte en `example_config_with0dash_with1uppercase`.
4. Sustituya todas las letras minúsculas por letras mayúsculas.
5. Añada el prefijo `LIVY_` al nombre de la variable.
6. Use el siguiente comando para habilitar la configuración de Livy. *`YOUR_VARIABLE_NAME`*.valor=*`yourvalue`*

Por ejemplo, para establecer las configuraciones de `livy.server.recovery.state-store = filesystem` y `spark.kubernetes.executor.podNamePrefix = my-prefix`, use estas propiedades de Helm:

```
-set env.LIVY_LIVY_SERVER_RECOVERY_STATE0STORE.value=filesystem
-set env.LIVY_SPARK_KUBERNETES_EXECUTOR_POD0NAME0PREFIX.value=myprefix
```

Administración de las ejecuciones de trabajos de Amazon EMR en EKS

En las siguientes secciones, se tratan temas que le ayudan a administrar las ejecuciones de trabajos de Amazon EMR en EKS. Estas incluyen configurar los parámetros de ejecución de los trabajos cuando los usas AWS CLI, configurar cómo se almacenan los datos de registro, ejecutar scripts SQL de Spark para ejecutar consultas, entender los estados de ejecución de los trabajos y saber cómo monitorear los trabajos. Si quiere configurar y completar una ejecución de trabajos para procesar datos, puede trabajar en estos temas, por lo general en orden.

Temas

- [Administrar las ejecuciones de tareas con AWS CLI](#)
- [Ejecutar scripts SQL de StartJobRun Spark a través de la API](#)
- [Estados de ejecuciones de trabajos](#)
- [Visualización de trabajos en la consola de Amazon EMR](#)
- [Errores comunes al ejecutar trabajos](#)

Administrar las ejecuciones de tareas con AWS CLI

En este tema se explica cómo gestionar las ejecuciones de tareas con AWS Command Line Interface (AWS CLI). Aborda detalles sobre las propiedades, como los parámetros de seguridad, el controlador y varios ajustes de anulación. También incluye subtemas que abarcan diversas formas de configurar el registro.

Temas

- [Opciones para configurar una ejecución de trabajo](#)
- [Configure una ejecución de trabajo para utilizar registros de Amazon S3](#)
- [Configurar una ejecución de tareas para usar Amazon CloudWatch Logs](#)
- [Enumerar ejecuciones de trabajos](#)
- [Describir una ejecución de trabajo](#)
- [Cancelar una ejecución de trabajo](#)

Opciones para configurar una ejecución de trabajo

Utilice las siguientes opciones para configurar los parámetros de ejecución del trabajo:

- `--execution-role-arn`: debe proporcionar un rol de IAM que se utilice para ejecutar trabajos. Para obtener más información, consulte [Uso de roles de ejecución de trabajos con Amazon EMR en EKS](#).
- `--release-label`: puede implementar Amazon EMR en EKS con las versiones 5.32.0, 6.2.0 y posteriores de Amazon EMR. Amazon EMR en EKS no es compatible con las versiones de lanzamiento anteriores de Amazon EMR. Para obtener más información, consulte [Versiones de Amazon EMR en EKS](#).

- `--job-driver`: el controlador de trabajo se utiliza para proporcionar información sobre el trabajo principal. Se trata de un campo de tipo unión en el que solo puede pasar uno de los valores del tipo de trabajo que desee ejecutar. Los tipos de trabajo admitidos son:
 - Envío de trabajos de Spark: se usa para ejecutar un comando a través de `spark-submit`. Puedes usar este tipo de trabajo para ejecutar Scala, SparkR PySpark, SparkSQL y cualquier otro trabajo compatible a través de Spark Submit. Este tipo de trabajo tiene los siguientes parámetros:
 - Entrypoint: es la referencia del HCFS (sistema de archivos compatible con Hadoop) al archivo `jar/py` principal que quiere ejecutar.
 - EntryPointArguments - Esta es una serie de argumentos que quieres pasar a tu archivo `jar/py` principal. Debería manejar la lectura de estos parámetros mediante su código de punto de entrada. Cada argumento de la matriz debe estar separado por una coma. EntryPointArguments no puede contener corchetes o paréntesis, como `()`, `{}` o `[]`.
 - SparkSubmitParameters - Estos son los parámetros de chispa 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 `—conf` o `—class`. Para obtener más información, consulte [Launching Applications with spark-submit](#).
 - Trabajos de Spark SQL: se utilizan para ejecutar un archivo de consulta SQL a través de Spark SQL. Puede usar este tipo de trabajo para ejecutar trabajos de SparkSQL. Este tipo de trabajo tiene los siguientes parámetros:
 - Entrypoint: es la referencia del HCFS (sistema de archivos compatible con Hadoop) al archivo de consulta SQL que desea ejecutar.

Para ver una lista de parámetros de Spark adicionales que puede usar para un trabajo de Spark SQL, consulte [Ejecutar scripts SQL de StartJobRun Spark a través de la API](#).

- `--configuration-overrides`: puede anular las configuraciones predeterminadas de las aplicaciones suministrando un objeto de configuración. 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 se componen de la configuración que se desea anular en ese archivo. Es posible especificar varias clasificaciones para varias aplicaciones en un solo objeto JSON. Las clasificaciones de configuración disponibles varían según la versión de Amazon EMR. Para ver una lista de las clasificaciones de configuración que están disponibles para cada versión de Amazon EMR, consulte [Versiones de Amazon EMR en EKS](#).

Si pasa la misma configuración en una aplicación de anulación y en los parámetros de envío de Spark, prevalecerán los parámetros de envío de Spark. A continuación se muestra la lista completa de prioridades de configuración, en orden de mayor a menor.

- Configuración proporcionada al crear `SparkSession`.
- Configuración proporcionada como parte de `sparkSubmitParameters` mediante `–conf`.
- Configuración proporcionada como parte de las anulaciones de aplicaciones.
- Configuraciones optimizadas elegidas por Amazon EMR para la versión.
- Configuraciones de código abierto predeterminadas para la aplicación.

Para supervisar las ejecuciones de tareas mediante Amazon CloudWatch o Amazon S3, debe proporcionar los detalles de configuración de CloudWatch. Para obtener más información, consulte [Configure una ejecución de trabajo para utilizar registros de Amazon S3](#) y [Configurar una ejecución de tareas para usar Amazon CloudWatch Logs](#). Si el depósito o grupo de CloudWatch registros de S3 no existe, Amazon EMR lo crea antes de cargar los registros en el depósito.

- Para obtener una lista adicional de las opciones de configuración de Kubernetes, consulte [Propiedades de Spark en Kubernetes](#).

Las siguientes configuraciones de Spark no son compatibles.

- `spark.kubernetes.authenticate.driver.serviceAccountName`
- `spark.kubernetes.authenticate.executor.serviceAccountName`
- `spark.kubernetes.namespace`
- `spark.kubernetes.driver.pod.name`
- `spark.kubernetes.container.image.pullPolicy`
- `spark.kubernetes.container.image`

 Note

Puede utilizar `spark.kubernetes.container.image` para imágenes de Docker personalizadas. Para obtener más información, consulte [Personalización de las imágenes de Docker para Amazon EMR en EKS](#).

Configure una ejecución de trabajo para utilizar registros de Amazon S3

Para poder supervisar el progreso del trabajo y solucionar los errores, debe configurar los trabajos para que envíen la información de registro a Amazon S3, Amazon CloudWatch Logs o ambos. Este tema le ayuda a empezar a publicar registros de aplicaciones en Amazon S3 en los trabajos que se lanzan con Amazon EMR en EKS.

Política de IAM de los registros de S3

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

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

Note

Amazon EMR en EKS también puede crear un bucket de Amazon S3. Si no hay ningún bucket de Amazon S3 disponible, incluya el permiso “s3:CreateBucket” en la política de IAM.

Una vez que haya otorgado a su rol de ejecución los permisos adecuados para enviar registros a Amazon S3, los datos de registro se envían a las siguientes ubicaciones de Amazon S3 cuando

s3MonitoringConfiguration se transfiere a la sección monitoringConfiguration de una solicitud start-job-run, como se muestra en [Administrar las ejecuciones de tareas con AWS CLI](#).

- Registros del remitente://*virtual-cluster-id*/jobs/ /containers/*logUri*/(stderr.gz/stdout.gz*job-id*) *pod-name*
- Registros de controladores://*logUri*/*virtual-cluster-id*/jobs/ *job-id* /containers/ /spark-*spark-application-id* -driver/ *job-id* (stderr.gz/stdout.gz)
- Registros de ejecutores:*logUri*//*virtual-cluster-id*/jobs/ *job-id* /containers//*executor-pod-name*/(*spark-application-id*stderr.gz/stdout.gz)

Configurar una ejecución de tareas para usar Amazon CloudWatch Logs

Para supervisar el progreso de los trabajos y solucionar los errores, debe configurar los trabajos para que envíen la información de registro a Amazon S3, Amazon CloudWatch Logs o ambos. Este tema le ayuda a empezar a utilizar CloudWatch los registros en los trabajos que se lanzan con Amazon EMR en EKS. Para obtener más información sobre CloudWatch los registros, consulte [Supervisión de los archivos de registro](#) en la Guía del CloudWatch usuario de Amazon.

CloudWatch Registros: política de IAM

Para que sus trabajos envíen datos de registro a CloudWatch Logs, se deben incluir los siguientes permisos en la política de permisos del rol de ejecución del trabajo. Sustituya *my_log_group_name* y *my_log_stream_prefix* por los nombres del grupo de CloudWatch registros y los nombres del flujo de registros, respectivamente. Amazon EMR en EKS crea el grupo de registros y el flujo de registros si no existen, siempre y cuando el ARN del rol de ejecución tenga los permisos adecuados.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ],
}
```



```

    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
      ]
    }
  ]
}

```

Note

Amazon EMR en EKS también puede crear un flujo de registro. Si no existe ningún flujo de registro, la política de IAM debe incluir el permiso "logs:CreateLogGroup".

Una vez que hayas otorgado a tu función de ejecución los permisos adecuados, tu aplicación envía sus datos de registro a CloudWatch Logs cuando `cloudWatchMonitoringConfiguration` se los transfiere a la `monitoringConfiguration` sección de una `start-job-run` solicitud, como se muestra en la siguiente [Administrar las ejecuciones de tareas con AWS CLI](#).

En la `StartJobRun` API, `log_group_name` es el nombre del grupo de CloudWatch registros y `log_stream_prefix` el prefijo del nombre del flujo de CloudWatch registros. Puede consultar y buscar estos registros en la AWS Management Console.

- Registros del remitente: `logGroup//virtual-cluster-id/jobs/ logStreamPrefix / containers//(stderr/stdout)job-id) pod-name`
- Registros de controladores: `///jobs/ /containers/ /spark- -driver/ (stderr/stdout) logGroup logStreamPrefix virtual-cluster-id job-id spark-application-id job-id`
- Registros del ejecutor: `logGrouplogStreamPrefix/virtual-cluster-id/job-id/jobs/ spark-application-id /containers//executor-pod-name/(stderr/stdout)`

Enumerar ejecuciones de trabajos

Puede ejecutar `list-job-run` para mostrar los estados de las ejecuciones de los trabajos, como se muestra en el siguiente ejemplo.

```
aws emr-containers list-job-runs --virtual-cluster-id <cluster-id>
```

Describir una ejecución de trabajo

Puede ejecutar `describe-job-run` para obtener más detalles sobre el trabajo, como el estado, los detalles del estado y el nombre del trabajo, tal como se muestra en el siguiente ejemplo.

```
aws emr-containers describe-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Cancelar una ejecución de trabajo

Puede ejecutar `cancel-job-run` para cancelar los trabajos en ejecución, tal como se muestra en el siguiente ejemplo.

```
aws emr-containers cancel-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Ejecutar scripts SQL de StartJobRun Spark a través de la API

Las versiones 6.7.0 y posteriores de Amazon EMR en EKS incluyen un controlador de tareas de Spark SQL para que pueda ejecutar scripts de Spark SQL a través de la API `StartJobRun`. Puede suministrar archivos de punto de entrada de SQL para ejecutar directamente consultas de Spark SQL en Amazon EMR en EKS con la API de `StartJobRun`, sin modificar los scripts de Spark SQL existentes. En la siguiente tabla, se enumeran los parámetros de Spark compatibles con los trabajos de Spark SQL a través de la `StartJobRun` API.

Puede elegir entre los siguientes parámetros de Spark para enviarlos a un trabajo de Spark SQL. Use estos parámetros para anular las propiedades predeterminadas de Spark.

Opción	Descripción
<code>--name NAME</code>	Nombre de la aplicación
<code>--jars JARS</code>	Lista de archivos jar separados por comas que se incluirán con el classpath de controlador y ejecutor.

Opción	Descripción
--packages	Lista de coordenadas Maven de los archivos jar separadas por comas para incluirlas en los classpaths de controlador y ejecutor.
--exclude-packages	Lista separada por comas de groupId:artifactId, para excluir mientras se resuelven las dependencias proporcionadas en --packages para evitar conflictos de dependencia.
--repositories	Lista de repositorios remotos adicionales separados por comas para buscar las coordenadas Maven proporcionadas con --packages.
--files FILES	Lista de archivos separados por comas que se colocarán en el directorio de trabajo de cada ejecutor.
--conf PROP=VALUE	Propiedad de configuración de Spark.
--properties-file FILE	Ruta a un archivo desde el que cargar propiedades adicionales.
--driver-memory MEM	Memoria para el controlador. 1024 MB por defecto.
--driver-java-options	Opciones de Java adicionales para pasarlas al controlador.
--driver-library-path	Entradas adicionales de la ruta de la biblioteca para pasarlas al controlador.
--driver-class-path	Entradas de ruta de clases adicionales para pasarlas al controlador.
--executor-memory MEM	Memoria por ejecutor. 1 GB por defecto.

Opción	Descripción
<code>--driver-cores NUM</code>	Número de núcleos utilizados por el controlador.
<code>--total-executor-cores NUM</code>	Núcleos totales para todos los ejecutores.
<code>--executor-cores NUM</code>	Número de núcleos utilizados por cada ejecutor.
<code>--num-executors NUM</code>	Número de ejecutores que se van a lanzar.
<code>-hivevar <key=value></code>	Sustitución de variables para aplicarla a los comandos de Hive, por ejemplo, <code>-hivevar A=B</code>
<code>-hiveconf <property=value></code>	Valor que se va a usar para la propiedad dada.

Para un trabajo de Spark SQL, cree un `start-job-run-request` archivo.json y especifique los parámetros necesarios para la ejecución del trabajo, como en el siguiente ejemplo:

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.7.0-latest",
  "jobDriver": {
    "sparkSqlJobDriver": {
      "entryPoint": "entryPoint_location",
      "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "2G"
        }
      }
    ]
  }
}
```

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

Estados de ejecuciones de trabajos

Al enviar una ejecución de trabajo a una cola de trabajos de Amazon EMR en EKS, el trabajo pasa al estado PENDING. 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 ejecuciones de trabajos pueden tener los siguientes estados:

- **PENDING:** el estado inicial del trabajo cuando la ejecución del trabajo se envía a Amazon EMR en EKS. El trabajo está a la espera de enviarse al clúster virtual y Amazon EMR en EKS está trabajando para enviarlo.
- **SUBMITTED:** una ejecución de trabajo que se envió correctamente al clúster virtual. A continuación, el programador de clústeres intenta ejecutar este trabajo en el clúster.
- **RUNNING:** una ejecución de trabajo que se está ejecutando en el clúster virtual. En las aplicaciones de Spark, esto significa que el estado del proceso del controlador de Spark es `running`.
- **FAILED:** una ejecución de trabajo que no se pudo enviar al clúster virtual o que se completó sin éxito. Consulta `StateDetails` y `FailureReason` encuentra información adicional sobre este error en el trabajo.
- **COMPLETED:** una ejecución de trabajo que se ha completado correctamente.
- **CANCEL_PENDING:** se ha solicitado la cancelación de una ejecución de trabajo. Amazon EMR en EKS está intentando cancelar el trabajo en el clúster virtual.
- **CANCELLED:** una ejecución de trabajo que se canceló correctamente.

Visualización de trabajos en la consola de Amazon EMR

Los datos de ejecución de los trabajos están disponibles para su visualización, por lo que puede monitorizar cada trabajo a medida que pasa por los estados. Para ver trabajos en la consola de Amazon EMR, siga estos pasos.

1. En el menú de la izquierda de la consola de Amazon EMR, en Amazon EMR en EKS, elija Clústeres virtuales.
2. En la lista de clústeres virtuales, seleccione el clúster virtual del que desee ver los trabajos.
3. En la tabla Ejecuciones de tareas, seleccione Ver registros para ver los detalles de una ejecución de trabajos.

Note

La compatibilidad con la experiencia de un solo clic está habilitada de forma predeterminada. Se puede desactivar al configurar `persistentAppUI` a `DISABLED` en `monitoringConfiguration` durante el envío del trabajo. Para obtener más información, consulte [Ver interfaces de usuario de aplicaciones persistentes](#).

Errores comunes al ejecutar trabajos

Se pueden producir los siguientes errores al ejecutar la API de `StartJobRun`. En la tabla se muestra cada error y se proporcionan los pasos de mitigación para que pueda abordar los problemas rápidamente.

Mensaje de error	Condición de error	Siguiente paso recomendado
error: argumento -- <i>argument</i> es obligatorio	Faltan parámetros obligatorios.	Agregue los argumentos que faltan a la solicitud de la API.
Se produjo un error (AccessDeniedException) al llamar a la StartJobRun operación: el usuario: no <i>ARN</i> está autorizado a realizar: emr-containers: StartJobRun	Falta el rol de ejecución.	Consulte Uso de Uso de roles de ejecución de trabajos con Amazon EMR en EKS .

Errores comunes de ejecución de tareas

207

Mensaje de error	Condición de error	Siguiente paso recomendado
Se produjo un error (ValidationException) al llamar a la StartJobRun operación: el estado del clúster virtual no state es válido para crear un recurso JobRun.	El clúster virtual no está preparado para ejecutar el trabajo.	Consulte Estados del clúster virtual .
Se produjo un error (ResourceNotFoundException) al llamar a la StartJobRun operación: la versión RELEASE no existe.	La versión especificada en el envío del trabajo es incorrecta.	Consulte Versiones de Amazon EMR en EKS .
Se produjo un error (AccessDeniedException) al llamar a la StartJobRun operación: el usuario: no ARN está autorizado a realizar: emr-containers: StartJobRun on resource: ARN con una denegación explícita. Se produjo un error (AccessDeniedException) al llamar a la StartJobRun operación: el usuario: no ARN está autorizado a realizar: emr-containers: on resource: StartJobRun ARN	El usuario no está autorizado a llamar. StartJobRun	Consulte Uso de roles de ejecución de trabajos con Amazon EMR en EKS .

Mensaje de error	Condición de error	Siguiente paso recomendado
Se produjo un error (ValidationException) al llamar a la StartJobRun operación : ConfigurationOverrides.MonitoringConfiguration.s3 MonitoringConfiguration .LogUri no pudo cumplir la restricción: %s	La sintaxis del URI de la ruta de S3 no es válida.	logURI debe tener el formato s3://...

Se pueden producir los siguientes errores al ejecutar la API de DescribeJobRun antes de que se ejecute el trabajo.

Mensaje de error	Condición de error	Siguiente paso recomendado
JobRun StateDetails: error en el envío. <i>classification</i> No se admite la clasificación. failureReason: VALIDATION_ERROR state: FAILED.	Los parámetros StartJobRun incluidos no son válidos.	Consulte Versiones de Amazon EMR en EKS .
StateDetails: <i>EKS Cluster ID</i> el clúster no existe. failureReason: CLUSTER_UNAVAILABLE state: FAILED	El clúster de EKS no está disponible.	Compruebe si el clúster de EKS existe y tiene los permisos correctos. Para obtener más información, consulte Configuración de Amazon EMR en EKS .
StateDetails: <i>EKS Cluster ID</i> El clúster no tiene permisos suficientes.	Amazon EMR no tiene permisos para acceder al clúster de EKS.	Compruebe que los permisos estén configurados para Amazon EMR en el espacio

Mensaje de error	Condición de error	Siguiente paso recomendado
failureReason: CLUSTER_UNAVAILABLE state: FAILED		de nombres registrado. Para obtener más información, consulte Configuración de Amazon EMR en EKS.
StateDetails: actualmente no <i>EKS Cluster ID</i> se puede acceder al clúster. failureReason: CLUSTER_UNAVAILABLE state: FAILED	No se puede llegar al clúster de EKS.	Compruebe si el clúster de EKS existe y tiene los permisos correctos. Para obtener más información, consulte Configuración de Amazon EMR en EKS.
StateDetails: el JobRun envió falló debido a un error interno. failureReason: INTERNAL_ERROR state: FAILED	Se produjo un error interno en el clúster de EKS.	N/A
StateDetails: <i>EKS Cluster ID</i> El clúster no tiene recursos suficientes. failureReason: USER_ERROR state: FAILED	No hay recursos suficientes en el clúster de EKS para ejecutar el trabajo.	Agregue más capacidad al grupo de nodos de EKS o configure el escalador automático de EKS. Para obtener más información, consulte Escalador automático de clústeres.

Se pueden producir los siguientes errores al ejecutar la API de DescribeJobRun tras la ejecución del trabajo.

Mensaje de error	Condición de error	Siguiente paso recomendado
StateDetails: Problemas para monitorear su. JobRun <i>EKS Cluster ID</i> El clúster no existe. failureReason: CLUSTER_UNAVAILABLE state: FAILED	El clúster de EKS no existe.	Compruebe si el clúster de EKS existe y tiene los permisos correctos. Para obtener más información, consulte Configuración de Amazon EMR en EKS .
StateDetails: Problemas para monitorear su. JobRun <i>EKS Cluster ID</i> El clúster no tiene permisos suficientes. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Amazon EMR no tiene permisos para acceder al clúster de EKS.	Compruebe que los permisos estén configurados para Amazon EMR en el espacio de nombres registrado. Para obtener más información, consulte Configuración de Amazon EMR en EKS .
StateDetails: Problemas para monitorear su. JobRun No <i>EKS Cluster ID</i> se puede acceder al clúster en este momento. failureReason: CLUSTER_UNAVAILABLE state: FAILED	No se puede llegar al clúster de EKS.	Compruebe si el clúster de EKS existe y tiene los permisos correctos. Para obtener más información, consulte Configuración de Amazon EMR en EKS .
StateDetails: Problemas para monitorizarlo JobRun debido a un error interno	Se ha producido un error interno que impide la JobRun supervisión.	N/A

Mensaje de error	Condición de error	Siguiente paso recomendado
failureReason: INTERNAL_ERROR state: FAILED		

El siguiente error puede producirse cuando un trabajo no puede iniciarse y el trabajo espera 15 minutos en el estado ENVIADO. Esto puede deberse a la falta de recursos del clúster.

Mensaje de error	Condición de error	Siguiente paso recomendado
tiempo de espera del clúster	El estado del trabajo ha sido ENVIADO durante 15 minutos o más.	Puede anular la configuración predeterminada de 15 minutos para este parámetro con la modificación de configuración que se muestra a continuación.

Use la siguiente configuración para cambiar el tiempo de espera del clúster a 30 minutos. Tenga en cuenta que proporciona el nuevo valor `job-start-timeout` en segundos:

```
{
  "configurationOverrides": {
    "applicationConfiguration": [{
      "classification": "emr-containers-defaults",
      "properties": {
        "job-start-timeout": "1800"
      }
    }]
  }
}
```

Uso de plantillas de trabajos

Una plantilla de trabajo almacena valores que se pueden compartir entre las invocaciones de la API `StartJobRun` al iniciar la ejecución de un trabajo. Admite dos casos de uso:

- Para evitar valores de solicitud de la API de `StartJobRun` periódicos y repetitivos.

- Para hacer cumplir una regla según la cual ciertos valores deben proporcionarse mediante solicitudes de la API de `StartJobRun`.

Las plantillas de trabajos permiten definir una plantilla reutilizable para las ejecuciones de trabajos a fin de aplicar una personalización adicional, por ejemplo:

- Configuración de la capacidad de computación del ejecutor y del controlador
- Establecer propiedades de seguridad y gobernanza, como los roles de IAM
- Personalizar una imagen de docker para usarla en múltiples aplicaciones y canalizaciones de datos

En los temas siguientes se proporciona información detallada sobre el uso de plantillas, incluido cómo usarlas para iniciar la ejecución de un trabajo y cómo cambiar los parámetros de las plantillas.

Temas

- [Creación y uso de una plantilla de trabajo para iniciar la ejecución de un trabajo](#)
- [Definición de parámetros de plantilla de trabajo](#)
- [Control del acceso a las plantillas de trabajos](#)

Creación y uso de una plantilla de trabajo para iniciar la ejecución de un trabajo

En esta sección se describe la creación de una plantilla de trabajo y su uso para iniciar la ejecución de un trabajo con AWS Command Line Interface (AWS CLI).

Para crear una nueva plantilla de trabajo

1. Cree un archivo `create-job-template-request.json` y especifique los parámetros necesarios para la plantilla de trabajo, como se muestra en el siguiente archivo JSON de ejemplo. Para obtener información sobre todos los parámetros disponibles, consulta la [CreateJobTemplateAPI](#).

La mayoría de los valores necesarios para la API `StartJobRun` también lo son para `jobTemplateData`. Si desea utilizar marcadores de posición para cualquier parámetro y proporcionar valores al invocar `StartJobRun` mediante una plantilla de trabajo, consulte la siguiente sección sobre los parámetros de la plantilla de trabajo.

```

{
  "name": "mytemplate",
  "jobTemplateData": {
    "executionRoleArn": "iam_role_arn_for_job_execution",
    "releaseLabel": "emr-6.7.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "entryPoint_location",
        "entryPointArguments": [ "argument1", "argument2", ... ],
        "sparkSubmitParameters": "--class <main_class> --conf
spark.executor.instances=2 --conf spark.executor.memory=2G --conf
spark.executor.cores=2 --conf spark.driver.cores=1"
      }
    },
    "configurationOverrides": {
      "applicationConfiguration": [
        {
          "classification": "spark-defaults",
          "properties": {
            "spark.driver.memory": "2G"
          }
        }
      ],
      "monitoringConfiguration": {
        "persistentAppUI": "ENABLED",
        "cloudWatchMonitoringConfiguration": {
          "logGroupName": "my_log_group",
          "logStreamNamePrefix": "log_stream_prefix"
        },
        "s3MonitoringConfiguration": {
          "logUri": "s3://my_s3_log_location/"
        }
      }
    }
  }
}

```

- Utilice el comando `create-job-template` con una ruta al archivo `create-job-template-request.json` almacenado localmente.

```

aws emr-containers create-job-template \
--cli-input-json file:///./create-job-template-request.json

```

Para iniciar un trabajo con una plantilla de trabajo

Especifique el ID del clúster virtual, el ID de la plantilla de trabajo y el nombre del trabajo en el comando `StartJobRun`, tal como se muestra en el siguiente ejemplo.

```
aws emr-containers start-job-run \  
--virtual-cluster-id 123456 \  
--name myjob \  
--job-template-id 1234abcd
```

Definición de parámetros de plantilla de trabajo

Los parámetros de la plantilla de trabajo le permiten especificar variables en la plantilla de trabajo. Los valores de estas variables de parámetros deberán especificarse al iniciar la ejecución de un trabajo con esa plantilla de trabajo. Los parámetros de la plantilla de trabajo se especifican en formato `${parameterName}`. Puede optar por especificar cualquier valor en un campo `jobTemplateData` como parámetro de la plantilla de trabajo. Para cada una de las variables de los parámetros de la plantilla de trabajo, especifique su tipo de datos (STRING o NUMBER) y, si lo desea, un valor por defecto. En el siguiente ejemplo, se muestra cómo especificar los parámetros de la plantilla de trabajo para los valores de ubicación del punto de entrada, clase principal y ubicación de registro de S3.

Para especificar la ubicación del punto de entrada, la clase principal y la ubicación del registro de Amazon S3 como parámetros de la plantilla de trabajo

1. Cree un archivo `create-job-template-request.json` y especifique los parámetros necesarios para la plantilla de trabajo, como se muestra en el siguiente archivo JSON de ejemplo. Para obtener más información sobre los parámetros, consulte la [CreateJobTemplateAPI](#).

```
{  
  "name": "mytemplate",  
  "jobTemplateData": {  
    "executionRoleArn": "iam_role_arn_for_job_execution",  
    "releaseLabel": "emr-6.7.0-latest",  
    "jobDriver": {  
      "sparkSubmitJobDriver": {  
        "entryPoint": "${EntryPointLocation}",  
        "entryPointArguments": [ "argument1", "argument2", ... ],  
        "sparkSubmitParameters": "--class ${MainClass} --conf  
spark.executor.instances=2 --conf spark.executor.memory=2G --conf  
spark.executor.cores=2 --conf spark.driver.cores=1"
```

```

    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "2G"
        }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "${LogS3BucketUri}"
      }
    }
  },
  "parameterConfiguration": {
    "EntryPointLocation": {
      "type": "STRING"
    },
    "MainClass": {
      "type": "STRING",
      "defaultValue": "Main"
    },
    "LogS3BucketUri": {
      "type": "STRING",
      "defaultValue": "s3://my_s3_log_location/"
    }
  }
}

```

- Utilice el comando `create-job-template` con una ruta al archivo `create-job-template-request.json` almacenado localmente o en Amazon S3.

```

aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json

```


Para iniciar un trabajo mediante una plantilla de trabajo con los parámetros de la plantilla de trabajo

Para iniciar la ejecución de un trabajo con una plantilla de trabajo que contenga los parámetros de la plantilla de trabajo, especifique el identificador de la plantilla de trabajo y los valores de los parámetros de la plantilla de trabajo en la solicitud de la API de `StartJobRun`, tal como se muestra a continuación.

```
aws emr-containers start-job-run \  
--virtual-cluster-id 123456 \  
--name myjob \  
--job-template-id 1234abcd \  
--job-template-parameters '{"EntryPointLocation": "entry_point_location", "MainClass":  
"ExampleMainClass", "LogS3BucketUri": "s3://example_s3_bucket/"}'
```

Control del acceso a las plantillas de trabajos

La política de `StartJobRun` le permite imponer que un usuario o un rol solo pueda ejecutar trabajos mediante las plantillas de trabajo que usted especifique y no puede ejecutar operaciones `StartJobRun` sin usar las plantillas de trabajo especificadas. Para ello, primero asegúrese de conceder al usuario o rol un permiso de lectura para las plantillas de trabajos especificadas, tal y como se muestra a continuación.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "emr-containers:DescribeJobTemplate",  
      "Resource": [  
        "job_template_1_arn",  
        "job_template_2_arn",  
        ...  
      ]  
    }  
  ]  
}
```

Para garantizar que un usuario o rol solo pueda invocar una operación `StartJobRun` cuando utilice plantillas de trabajo específicas, puede asignar el siguiente permiso de política de `StartJobRun` a un usuario o rol determinado.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "virtual_cluster_arn",
      ],
      "Condition": [
        "StringEquals": {
          "emr-containers:JobTemplateArn": [
            "job_template_1_arn",
            "job_template_2_arn",
            ...
          ]
        }
      ]
    }
  ]
}
```

Si la plantilla de trabajo especifica un parámetro de plantilla de trabajo dentro del campo ARN del rol de ejecución, el usuario podrá proporcionar un valor para este parámetro y, por lo tanto, podrá invocar `StartJobRun` mediante un rol de ejecución arbitrario. Para restringir los roles de ejecución que el usuario puede proporcionar, consulte [Controlar el acceso al rol de ejecución en Uso de roles de ejecución de trabajos con Amazon EMR en EKS](#).

Si no se especifica ninguna condición en la política de acción `StartJobRun` anterior para un usuario o rol determinado, el usuario o el rol podrán invocar una acción `StartJobRun` en el clúster virtual especificado con una plantilla de trabajo arbitraria a la que tengan acceso de lectura o mediante un rol de ejecución arbitrario.

Uso de plantillas de pods

A partir de las versiones 5.33.0 o 6.3.0 de Amazon EMR, Amazon EMR en EKS es compatible con la característica de plantillas de pods de Spark. Un pod es un grupo de uno o más contenedores, con recursos de red y almacenamiento compartidos, y una especificación sobre cómo ejecutar los contenedores. Las plantillas de pods son especificaciones que determinan cómo ejecutar cada pod.

Puede usar archivos de plantillas de pods para definir las configuraciones del pod de controlador o ejecutor que las configuraciones de Spark no admiten. Para obtener más información sobre la característica de plantillas de pods de Spark, consulte [Plantillas de pods](#).

 Note

La característica de plantillas de pods solo funciona con los pods controladores y ejecutores. No puede configurar los pods del remitente de trabajos mediante la plantilla de pods.

Escenarios habituales

Puede definir cómo ejecutar los trabajos de Spark en clústeres de EKS compartidos mediante plantillas de pods con Amazon EMR en EKS y ahorrar costos y mejorar la utilización y el rendimiento de los recursos.

- Para reducir los costes, puede programar las tareas del controlador de Spark para que se ejecuten en las instancias EC2 bajo demanda de Amazon y, al mismo tiempo, programar las tareas del ejecutor de Spark para que se ejecuten en las instancias EC2 puntuales de Amazon.
- Para aumentar la utilización de los recursos, puede ayudar a varios equipos a ejecutar sus cargas de trabajo en el mismo clúster de EKS. Cada equipo dispondrá de un grupo de EC2 nodos de Amazon designado en el que ejecutar sus cargas de trabajo. Puede utilizar plantillas de pods para aplicar la tolerancia correspondiente a su carga de trabajo.
- Para mejorar la supervisión, puede ejecutar un contenedor de registro independiente para reenviar los registros a la aplicación de supervisión existente.

Por ejemplo, el siguiente archivo de plantilla de pod muestra un escenario de uso común.

```
apiVersion: v1
kind: Pod
spec:
  volumes:
    - name: source-data-volume
      emptyDir: {}
    - name: metrics-files-volume
      emptyDir: {}
  nodeSelector:
    eks.amazonaws.com/nodegroup: emr-containers-nodegroup
```

```

containers:
- name: spark-kubernetes-driver # This will be interpreted as driver Spark main
  container
  env:
    - name: RANDOM
      value: "random"
  volumeMounts:
    - name: shared-volume
      mountPath: /var/data
    - name: metrics-files-volume
      mountPath: /var/metrics/data
- name: custom-side-car-container # Sidecar container
  image: <side_car_container_image>
  env:
    - name: RANDOM_SIDE CAR
      value: random
  volumeMounts:
    - name: metrics-files-volume
      mountPath: /var/metrics/data
  command:
    - /bin/sh
    - '-c'
    - <command-to-upload-metrics-files>
initContainers:
- name: spark-init-container-driver # Init container
  image: <spark-pre-step-image>
  volumeMounts:
    - name: source-data-volume # Use EMR predefined volumes
      mountPath: /var/data
  command:
    - /bin/sh
    - '-c'
    - <command-to-download-dependency-jars>

```

La plantilla de pod lleva a cabo las tareas siguientes:

- Agrega un nuevo [contenedor de inicialización](#) que se ejecuta antes de que se inicie el contenedor principal de Spark. El contenedor de inicio comparte el [EmptyDirvolumen](#) llamado source-data-volume con el contenedor principal de Spark. Puede hacer que su contenedor de inicialización ejecute los pasos de inicialización, como descargar dependencias o generar datos de entrada. A continuación, el contenedor principal de Spark consume los datos.

- Agregue otro [contenedor asociado](#) que se ejecute junto con el contenedor principal de Spark. Los dos contenedores comparten otro volumen EmptyDir llamado `metrics-files-volume`. Su trabajo en Spark puede generar métricas, como las métricas de Prometheus. A continuación, el trabajo de Spark puede colocar las métricas en un archivo y hacer que el contenedor asociado cargue los archivos en su propio sistema de BI para análisis futuros.
- Agregue una nueva variable de entorno al contenedor principal de Spark. Puede hacer que el trabajo consuma la variable de entorno.
- Defina un [selector de nodos](#) para que el pod solo esté programado en el grupo de nodos `emr-containers-nodegroup`. Esto ayuda a aislar los recursos de computación entre trabajos y equipos.

Habilitación de plantillas de pods con Amazon EMR en EKS

Para habilitar la característica de plantilla de pod con Amazon EMR en EKS, configure las propiedades de Spark `spark.kubernetes.driver.podTemplateFile` y `spark.kubernetes.executor.podTemplateFile` para que apunten a los archivos de plantillas de pods en Amazon S3. A continuación, Spark descarga el archivo de plantilla del pod y lo utiliza para crear los pods controladores y ejecutores.

Note

Spark usa el rol de ejecución de trabajos para cargar la plantilla de pod, por lo que el rol de ejecución de tareas debe tener permisos de acceso a Amazon S3 para cargar las plantillas de pods. Para obtener más información, consulte [Crear un rol de ejecución de trabajos](#).

Puede utilizar los `SparkSubmitParameters` para especificar la ruta de Amazon S3 a la plantilla del pod, tal como se muestra en el siguiente archivo JSON de ejecución de tareas.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
```

```

    "sparkSubmitParameters": "--class <main_class> \
      --conf
spark.kubernetes.driver.podTemplateFile=s3://path_to_driver_pod_template \
      --conf
spark.kubernetes.executor.podTemplateFile=s3://path_to_executor_pod_template \
      --conf spark.executor.instances=2 \
      --conf spark.executor.memory=2G \
      --conf spark.executor.cores=2 \
      --conf spark.driver.cores=1"
  }
}
}

```

Como alternativa, puede usar las `configurationOverrides` para especificar la ruta de Amazon S3 a la plantilla del pod, tal como se muestra en el siguiente archivo JSON de ejecución de tareas.

```

{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": [argument1, "argument2", ...],
      "sparkSubmitParameters": "--class <main_class> \
        --conf spark.executor.instances=2 \
        --conf spark.executor.memory=2G \
        --conf spark.executor.cores=2 \
        --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "2G",
          "spark.kubernetes.driver.podTemplateFile": "s3://path_to_driver_pod_template",
          "spark.kubernetes.executor.podTemplateFile": "s3://path_to_executor_pod_template"
        }
      }
    ]
  }
}

```

```
]
}
}
```

Note

1. Debe seguir las pautas de seguridad cuando utilice la característica de plantilla de pod con Amazon EMR en EKS, como aislar el código de una aplicación que no sea de confianza. Para obtener más información, consulte [Prácticas recomendadas de seguridad de Amazon EMR en EKS](#).
2. No puede cambiar los nombres de los contenedores principales de Spark mediante `spark.kubernetes.driver.podTemplateContainerName` y `spark.kubernetes.executor.podTemplateContainerName`, ya que estos nombres están codificados como `spark-kubernetes-driver` y `spark-kubernetes-executors`. Si quiere personalizar el contenedor principal de Spark, debe especificarlo en una plantilla de pod con estos nombres codificados.

Campos de plantilla de pod

Tenga en cuenta las siguientes restricciones de campo al configurar una plantilla de pod con Amazon EMR en EKS.

- Amazon EMR en EKS solo permite los siguientes campos en una plantilla de pod para habilitar una programación adecuada de los trabajos.

Estos son los campos de nivel de pod permitidos:

- `apiVersion`
- `kind`
- `metadata`
- `spec.activeDeadlineSeconds`
- `spec.affinity`
- `spec.containers`
- `spec.enableServiceLinks`
- `spec.ephemeralContainers`

- `spec.hostAliases`
- `spec.hostname`
- `spec.imagePullSecrets`
- `spec.initContainers`
- `spec.nodeName`
- `spec.nodeSelector`
- `spec.overhead`
- `spec.preemptionPolicy`
- `spec.priority`
- `spec.priorityClassName`
- `spec.readinessGates`
- `spec.runtimeClassName`
- `spec.schedulerName`
- `spec.subdomain`
- `spec.terminationGracePeriodSeconds`
- `spec.tolerations`
- `spec.topologySpreadConstraints`
- `spec.volumes`

Estos son los campos de nivel de contenedor principal de Spark permitidos:

- `env`
- `envFrom`
- `name`
- `lifecycle`
- `livenessProbe`
- `readinessProbe`
- `resources`
- `startupProbe`
- `stdin`

- `terminationMessagePath`

- `terminationMessagePolicy`
- `tty`
- `volumeDevices`
- `volumeMounts`
- `workingDir`

Cuando utiliza campos no permitidos en la plantilla de pod, Spark lanza una excepción y se produce un error en el trabajo. En el siguiente ejemplo, se muestra un mensaje de error en el registro del controlador de Spark porque hay campos no permitidos.

```
Executor pod template validation failed.  
Field container.command in Spark main container not allowed but specified.
```

- Amazon EMR en EKS predefine los siguientes parámetros en una plantilla de pod. Los campos que especifique en una plantilla de pod no deben superponerse con estos campos.

Estos son los nombres de volumen predefinidos:

- `emr-container-communicate`
- `config-volume`
- `emr-container-application-log-dir`
- `emr-container-event-log-dir`
- `temp-data-dir`
- `mnt-dir`
- `home-dir`
- `emr-container-s3`

Estos son los montajes de volumen predefinidos que solo se aplican al contenedor principal de Spark:

- Nombre:`emr-container-communicate`; MountPath `/var/log/fluentd`
- Nombre:`emr-container-application-log-dir`; MountPath: `/var/log/spark/user`
- Nombre:`emr-container-event-log-dir`; MountPath: `/var/log/spark/apps`
- Nombre:`mnt-dir`; MountPath: `/mnt`
- Nombre:`temp-data-dir`; MountPath: `/tmp`

- Nombre:`home-dir`; MountPath: `/home/hadoop`

Estas son las variables de entorno predefinidas que solo se aplican al contenedor principal de Spark:

- SPARK_CONTAINER_ID
- K8S_SPARK_LOG_URL_STDERR
- K8S_SPARK_LOG_URL_STDOUT
- SIDECAR_SIGNAL_FILE

 Note

Puede seguir usando estos volúmenes predefinidos y montarlos en sus contenedores asociados adicionales. Por ejemplo, puede usar `emr-container-application-log-dir` y montarlo en su propio contenedor asociado definido en la plantilla de pod.

Si los campos que especifique entran en conflicto con alguno de los campos predefinidos de la plantilla del pod, Spark lanza una excepción y se produce un error en el trabajo. En el siguiente ejemplo, se muestra un mensaje de error en el registro de la aplicación de Spark debido a conflictos con los campos predefinidos.

```
Defined volume mount path on main container must not overlap with reserved mount paths: [<reserved-paths>]
```

Consideraciones sobre los contenedores asociados

Amazon EMR controla el ciclo de vida de los pods aprovisionados por Amazon EMR en EKS. Los contenedores asociados deben seguir el mismo ciclo de vida que el contenedor principal de Spark. Si inyecta contenedores asociados adicionales en sus pods, le recomendamos que los integre con la administración del ciclo de vida de los pods que Amazon EMR define para que el contenedor asociado pueda detenerse solo cuando salga el contenedor principal de Spark.

Para reducir costos, le recomendamos que implemente un proceso que impida que los pods controladores con contenedores asociados sigan funcionando una vez que haya completado el trabajo. El controlador de Spark elimina los pods ejecutores cuando el ejecutor ha terminado. Sin embargo, cuando se completa un programa de controlador, los contenedores asociados adicionales siguen funcionando. El pod se factura hasta que Amazon EMR en EKS limpie el pod controlador,

normalmente en menos de un minuto después de que se complete el contenedor principal de Spark del controlador. Para reducir costos, puede integrar sus contenedores asociados adicionales con el mecanismo de administración del ciclo de vida que Amazon EMR en EKS define para los pods controladores y ejecutores, tal y como se describe en la siguiente sección.

El contenedor principal de Spark de los pods ejecutores y controladores envía heartbeat a un archivo `/var/log/fluentd/main-container-terminated` cada dos segundos. Al agregar el montaje de volumen `emr-container-communicate` predefinido de Amazon EMR a su contenedor asociado, puede definir un subproceso de dicho contenedor para hacer un seguimiento periódico de la hora de la última modificación de este archivo. A continuación, el subproceso se detiene automáticamente si descubre que el contenedor principal de Spark detiene el heartbeat durante más tiempo.

El siguiente ejemplo muestra un subproceso que rastrea el archivo de latidos y se detiene solo. *your_volume_mount* Sustitúyala por la ruta en la que se monta el volumen predefinido. El script está incluido dentro de la imagen utilizada por el contenedor asociado. En un archivo de plantilla de pod, puede especificar un contenedor asociado con los comandos `sub_process_script.sh` y `main_command`.

```
MOUNT_PATH="your_volume_mount"
FILE_TO_WATCH="$MOUNT_PATH/main-container-terminated"
INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD=60
HEARTBEAT_TIMEOUT_THRESHOLD=15
SLEEP_DURATION=10

function terminate_main_process() {
    # Stop main process
}

# Waiting for the first heartbeat sent by Spark main container
echo "Waiting for file $FILE_TO_WATCH to appear..."
start_wait=$(date +%s)
while ! [[ -f "$FILE_TO_WATCH" ]]; do
    elapsed_wait=$(expr $(date +%s) - $start_wait)
    if [ "$elapsed_wait" -gt "$INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD" ]; then
        echo "File $FILE_TO_WATCH not found after $INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD
seconds; aborting"
        terminate_main_process
        exit 1
    fi
    sleep $SLEEP_DURATION;
```

```
done;
echo "Found file $FILE_TO_WATCH; watching for heartbeats..."

while [[ -f "$FILE_TO_WATCH" ]]; do
    LAST_HEARTBEAT=$(stat -c %Y $FILE_TO_WATCH)
    ELAPSED_TIME_SINCE_AFTER_HEARTBEAT=$(expr $(date +%s) - $LAST_HEARTBEAT)
    if [ "$ELAPSED_TIME_SINCE_AFTER_HEARTBEAT" -gt "$HEARTBEAT_TIMEOUT_THRESHOLD" ];
    then
        echo "Last heartbeat to file $FILE_TO_WATCH was more than
$HEARTBEAT_TIMEOUT_THRESHOLD seconds ago at $LAST_HEARTBEAT; terminating"
        terminate_main_process
        exit 0
    fi
    sleep $SLEEP_DURATION;
done;
echo "Outside of loop, main-container-terminated file no longer exists"

# The file will be deleted once the fluentd container is terminated

echo "The file $FILE_TO_WATCH doesn't exist any more;"
terminate_main_process
exit 0
```

Uso de políticas de reintento de trabajos

En Amazon EMR en EKS 6.9.0 y versiones posteriores, puede establecer una política de reintento para las ejecuciones de sus trabajos. Las políticas de reintento hacen que un pod controlador de tareas se reinicie automáticamente si se elimina o se produce un error. Esto hace que los trabajos de streaming Spark de larga duración sean más resilientes a errores.

Establecer una política de reintento para un trabajo

Para configurar una política de reintentos, proporciona un `RetryPolicyConfiguration` campo mediante la [StartJobRun](#) API. A continuación, se muestra un ejemplo de `retryPolicyConfiguration`:

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.9.0-latest \
```

```
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
    "entryPointArguments": [ "2" ],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --conf
spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
  }
}' \
--retry-policy-configuration '{
  "maxAttempts": 5
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

Note

`retryPolicyConfigurations` solo está disponible a partir de la AWS CLI versión 1.27.68. Para actualizarla AWS CLI a la última versión, consulte [Instalación o actualización de la última versión](#) de AWS CLI

Configure el campo `maxAttempts` con el número máximo de veces que desee que se reinicie el pod controlador de trabajos en caso de que se elimine o se produzca un error. El intervalo de ejecución entre dos intentos de reintento del controlador de tareas es un intervalo de reintento exponencial de (10 segundos, 20 segundos, 40 segundos...) limitado a 6 minutos, tal como se describe en la [documentación de Kubernetes](#).

Note

Cada ejecución adicional del controlador de trabajos se facturará como otra ejecución de tareas y estará sujeta a los [precios de Amazon EMR en EKS](#).

Vuelva a intentar los valores de configuración de la política

- Política de reintento predeterminada para un trabajo: `StartJobRun` incluye una política de reintento establecida en un intento máximo de forma predeterminada. Puede configurar la política de reintento como desee.

Note

Si `maxAttempts` de `retryPolicyConfiguration` se establece en 1, significa que no se llevará a cabo ningún reintento para que aparezca el pod controlador en caso de error.

- Inhabilitar la política de reintentos para un trabajo: para deshabilitar una política de reintentos, establezca el valor máximo de intentos en `retryPolicyConfiguration` 1.

```
"retryPolicyConfiguration": {  
  "maxAttempts": 1  
}
```

- Defina `maxAttempts` para un trabajo dentro del rango válido: la llamada de `StartJobRun` fallará si el valor `maxAttempts` está fuera del rango válido. El rango válido de `maxAttempts` es de 1 a 2 147 483 647 (entero de 32 bits), el rango admitido por los ajustes de configuración de Kubernetes `backOffLimit`. Para obtener más información, consulte [Política de retroceso de errores de pods](#) en la documentación de Kubernetes. Si el valor `maxAttempts` no es válido, se devuelve el siguiente mensaje de error:

```
{  
  "message": "Retry policy configuration's parameter value of maxAttempts is invalid"  
}
```

Recuperación del estado de una política de reintento de un trabajo

Puede ver el estado de los reintentos de un trabajo con la tecla y. [ListJobRunsDescribeJobRun](#) APIs Una vez que solicite un trabajo con una configuración de política de reintento habilitada, las respuestas `ListJobRun` y `DescribeJobRun` contendrán el estado de la política de reintentos en el campo `RetryPolicyExecution`. Además, la respuesta `DescribeJobRun` contendrá la `RetryPolicyConfiguration` que se ingresó en la solicitud de `StartJobRun` del trabajo.

Respuestas de ejemplo

ListJobRuns response

```
{
  "jobRuns": [
    ...
    ...
    "retryPolicyExecution" : {
      "currentAttemptCount": 2
    }
    ...
    ...
  ]
}
```

DescribeJobRun response

```
{
  ...
  ...
  "retryPolicyConfiguration": {
    "maxAttempts": 5
  },
  "retryPolicyExecution" : {
    "currentAttemptCount": 2
  },
  ...
  ...
}
```

Estos campos no estarán visibles cuando la política de reintento esté deshabilitada en el trabajo, tal como se describe en [Vuelva a intentar los valores de configuración de la política](#).

Supervisión de un trabajo con una política de reintento

Al activar una política de reintentos, se genera un CloudWatch evento para cada controlador de trabajo que se cree. Para suscribirse a estos eventos, configure una regla de CloudWatch eventos mediante el siguiente comando:

```
aws events put-rule \
--name cwe-test \
```

```
--event-pattern '{"detail-type": ["EMR Job Run New Driver Attempt"]}'
```

El evento devolverá información sobre el `newDriverPodName`, la marca de tiempo de `newDriverCreatedAt`, `previousDriverFailureMessage` y los `currentAttemptCount` de los controladores del trabajo. Estos eventos no se crearán si la política de reintento está deshabilitada.

Para obtener más información sobre cómo supervisar su trabajo con CloudWatch eventos, consulte [Supervisa los trabajos con Amazon CloudWatch Events](#).

Búsqueda de registros para controladores y ejecutores

Los nombres de los pods controladores siguen el formato `spark-<job id>-driver-<random-suffix>`. El mismo `random-suffix` se agrega a los nombres de los pods ejecutores que genera el controlador. Al usar este `random-suffix`, puede buscar los registros de un controlador y sus ejecutores asociados. El `random-suffix` solo está presente si la [política de reintento está habilitada](#) para el trabajo; de lo contrario, el `random-suffix` está ausente.

Para obtener más información sobre cómo configurar los trabajos con la configuración de supervisión del registro, consulte [Ejecutar una aplicación de Spark](#).

Uso de la rotación del registro de eventos de Spark

Con Amazon EMR 6.3.0 y versiones posteriores, puede activar la característica de rotación del registro de eventos de Spark de Amazon EMR en EKS. En lugar de generar un único archivo de registro de eventos, esta característica rota el archivo en función del intervalo de tiempo configurado y elimina los archivos de registro de eventos más antiguos.

La rotación de registros de eventos de Spark puede ayudarle a evitar posibles problemas con un archivo de registro de eventos de Spark de gran tamaño que se genera para trabajos de larga duración o en streaming. Por ejemplo, empieza un trabajo de Spark de larga duración con un registro de eventos activado con el parámetro `persistentAppUI`. El controlador de Spark genera un archivo de registro de eventos. Si el trabajo se ejecuta durante horas o días y el espacio en disco en el nodo de Kubernetes es limitado, el archivo de registro de evento puede consumir todo el espacio disponible en el disco. Activar la característica de rotación del registro de eventos de Spark resuelve el problema al dividir el archivo de registro en varios archivos y eliminar los archivos más antiguos.

 Note

Esta característica solo funciona con Amazon EMR en EKS. Amazon EMR que se ejecuta en Amazon EC2 no admite la rotación del registro de eventos de Spark.

Para activar la característica de rotación del registro de eventos de Spark, configure los siguientes parámetros de Spark:

- `spark.eventLog.rotation.enabled`: activa la rotación del registro. Está deshabilitado de forma predeterminada en el archivo de configuración de Spark. Configúrelo en verdadero para activar la característica.
- `spark.eventLog.rotation.interval`: especifica el intervalo de tiempo para la rotación del registro. El valor mínimo es de 60 segundos. El valor de predeterminado es de 300 segundos.
- `spark.eventLog.rotation.minFileSize`: especifica un tamaño de archivo mínimo para rotar el archivo de registro. El valor mínimo y predeterminado es de 1 MB.
- `spark.eventLog.rotation.maxFilesToRetain`: especifica cuántos archivos de registro rotados se deben conservar durante la limpieza. El rango válido es de 1 a 10. El valor predeterminado es 2.

Puede especificar estos parámetros en la sección `sparkSubmitParameters` de la API [StartJobRun](#), tal como se muestra en el siguiente ejemplo.

```
"sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf  
spark.eventLog.rotation.enabled=true --conf spark.eventLog.rotation.interval=300 --  
conf spark.eventLog.rotation.minFileSize=1m --conf  
spark.eventLog.rotation.maxFilesToRetain=2"
```

Uso de la rotación de los registros de contenedores de Spark

Con Amazon EMR 6.11.0 y versiones posteriores, puede activar la característica de rotación de los registros de contenedores de Spark para Amazon EMR en EKS. En lugar de generar un único archivo `stdout` o un archivo de registro `stderr`, esta característica rota el archivo en función del tamaño de rotación configurado y elimina los archivos de registro más antiguos del contenedor.

Rotar los registros de un contenedor de Spark puede ayudarle a evitar posibles problemas con los archivos de registro de Spark de gran tamaño que se generan para trabajos de larga duración o

en streaming. Por ejemplo, puede iniciar un trabajo de Spark de larga duración, y el controlador de Spark genera un archivo de registro de contenedor. Si el trabajo se ejecuta durante horas o días y el espacio en disco en el nodo de Kubernetes es limitado, el archivo de registro de contenedor puede consumir todo el espacio disponible en el disco. Al activar la rotación de los registros de contenedor de Spark, se divide el archivo de registro en varios archivos y se eliminan los archivos más antiguos.

Para activar la característica de rotación de los registros de contenedores de Spark, configure los siguientes parámetros de Spark:

containerLogRotationConfiguration

Incluya este parámetro en `monitoringConfiguration` para activar la rotación del registro. Está deshabilitado de forma predeterminada. Debe usar `containerLogRotationConfiguration` además de `s3MonitoringConfiguration`.

rotationSize

El parámetro `rotationSize` especifica el tamaño del archivo para la rotación del registro. El rango de valores posibles va de 2KB a 2GB. La parte de unidades numéricas del parámetro `rotationSize` se pasa como un número entero. Como no se admiten valores decimales, puede especificar un tamaño de rotación de 1,5 GB, por ejemplo, con el valor `1500MB`.

maxFilesToKeep

El parámetro `maxFilesToKeep` especifica el número máximo de archivos que deben retenerse en el contenedor después de que se haya completado la rotación. El valor mínimo es 1 y el máximo, 50.

Puede especificar estos parámetros en la sección `monitoringConfiguration` de la API `StartJobRun`, tal como se muestra en el siguiente ejemplo. En este ejemplo, con `rotationSize` = "10 MB" y `maxFilesToKeep` = 3, Amazon EMR en EKS rota los registros a 10 MB, genera un nuevo archivo de registro y, a continuación, purga el archivo de registro más antiguo cuando el número de archivos de registro llega a 3.

```
{
  "name": "my-long-running-job",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
```

```

    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
      "sparkSubmitParameters": "--class main_class --conf spark.executor.instances=2
--conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "2G"
        }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      },
      "containerLogRotationConfiguration": {
        "rotationSize": "10MB",
        "maxFilesToKeep": "3"
      }
    }
  }
}

```

Para iniciar la ejecución de un trabajo con la rotación del registro del contenedor de Spark, incluya en el comando una ruta al archivo JSON que configuró en el comando [StartJobRun](#).

```

aws emr-containers start-job-run \
--cli-input-json file://path-to-json-request-file

```

Uso del escalado automático vertical con trabajos de Spark de Amazon EMR

El escalado automático vertical de Amazon EMR en EKS ajusta automáticamente los recursos de memoria y CPU para adaptarlos a las necesidades de la carga de trabajo que proporciona a las aplicaciones de Spark de Amazon EMR. Esto simplifica la administración de recursos.

Para hacer un seguimiento del historial de uso y del uso en tiempo real de los recursos de sus aplicaciones de Spark de Amazon EMR, el escalado automático vertical aprovecha el [escalador automático vertical de pods \(VPA\)](#) de Kubernetes. La capacidad de escalado automático vertical utiliza los datos que recopila el VPA para ajustar automáticamente los recursos de memoria y CPU asignados a las aplicaciones de Spark. Este proceso simplificado mejora la fiabilidad y optimiza los costos.

Temas

- [Configuración del escalado automático vertical de Amazon EMR en EKS](#)
- [Cómo empezar a utilizar el escalado automático vertical de Amazon EMR en EKS](#)
- [Configuración del escalado automático vertical de Amazon EMR en EKS](#)
- [Supervisión del escalado automático vertical de Amazon EMR en EKS](#)
- [Desinstalar el operador de escalado automático vertical de Amazon EMR en EKS](#)

Configuración del escalado automático vertical de Amazon EMR en EKS

Este tema le ayuda a preparar su clúster de Amazon EKS para enviar trabajos de Spark de Amazon EMR con escalado automático vertical. El proceso de configuración requiere que confirme o complete las tareas de las siguientes secciones:

Temas

- [Requisitos previos](#)
- [Instalar Operator Lifecycle Manager \(OLM\) en su clúster de Amazon EKS](#)
- [Instalar el operador de escalado automático vertical de Amazon EMR en EKS](#)

Requisitos previos

Complete las siguientes tareas antes de instalar el operador de Kubernetes con escalado automático vertical en el clúster. Si ya ha completado alguno de los requisitos previos, puede omitirlos y pasar al siguiente.

- [Instalar o actualizar a la última versión de AWS CLI](#): si ya ha instalado el AWS CLI, confirme que tiene la última versión.
- [Instale kubectl](#): es una herramienta de línea de comandos que se utiliza para comunicarse con el servidor de la API de Kubernetes. Necesita kubectl para instalar y supervisar los artefactos relacionados con el escalado automático vertical en su clúster de Amazon EKS.
- [Instale Operator SDK](#): Amazon EMR en EKS utiliza Operator SDK como administrador de paquetes durante toda la vida útil del operador de escalado automático vertical que instale en el clúster.
- [Instale Docker](#): necesita acceso a la CLI de Docker para autenticar y obtener las imágenes de Docker verticales relacionadas con el escalado automático para instalarlas en su clúster de Amazon EKS.
- [Instale el servidor de métricas de Kubernetes](#): primero debe instalar el servidor de métricas para que el escalador automático del pod vertical pueda obtener las métricas del servidor de la API de Kubernetes.
- [Comience con Amazon EKS: eksctl](#) (versión 1.24 o posterior): el escalado automático vertical es compatible con las versiones 1.24 y posteriores de Amazon EKS. Una vez creado el clúster, [regístrelo para usarlo en Amazon EMR](#).
- [Seleccione un URI de imagen base de Amazon EMR](#) (versión 6.10.0 o posterior): el escalado automático vertical es compatible con las versiones 6.10.0 y posteriores de Amazon EMR.

Instalar Operator Lifecycle Manager (OLM) en su clúster de Amazon EKS

Utilice la CLI de Operator SDK para instalar Operator Lifecycle Manager (OLM) en el clúster de Amazon EMR en EKS en el que desee configurar el escalado automático vertical, tal como se muestra en el siguiente ejemplo. Una vez que lo haya configurado, puede usar OLM para instalar y administrar el ciclo de vida del [operador de escalado automático vertical de Amazon EMR](#).

```
operator-sdk olm install
```

Para validar la instalación, ejecute el comando `olm status`:

```
operator-sdk olm status
```

Verifique que el comando devuelva un resultado correcto, similar al siguiente ejemplo:

```
INFO[0007] Successfully got OLM status for version X.XX
```

Si la instalación no se lleva a cabo correctamente, consulte [Solución de problemas en el escalado automático vertical de Amazon EMR en EKS](#).

Instalar el operador de escalado automático vertical de Amazon EMR en EKS

Siga estos pasos para instalar el operador de escalado automático vertical en su clúster de Amazon EKS:

1. Configure las siguientes variables de entorno que utilizará para completar la instalación:
 - **\$REGION** apunta a la Región de AWS de su clúster. Por ejemplo, `us-west-2`.
 - **\$ACCOUNT_ID** apunta al ID de cuenta de Amazon ECR de su región. Para obtener más información, consulte [Cuentas de registro de Amazon ECR por región](#).
 - **\$RELEASE** apunta a la versión de Amazon EMR que desea usar para su clúster. Con el escalado automático vertical, debe utilizar la versión 6.10.0 o una posterior de Amazon EMR.
2. A continuación, ingrese los tokens de autenticación del operador en el [registro de Amazon ECR](#).

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

3. Instale Amazon EMR en EKS en el operador de escalado automático vertical con el siguiente comando:

```
ECR_URL=$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com && \
REPO_DEST=dynamic-sizing-k8s-operator-olm-bundle && \
BUNDLE_IMG=emr-$RELEASE-dynamic-sizing-k8s-operator && \
operator-sdk run bundle \
$ECR_URL/$REPO_DEST/$BUNDLE_IMG\:latest
```

Esto creará una versión del operador de escalado automático vertical en el espacio de nombres predeterminado del clúster de Amazon EKS. Utilice este comando para llevar a cabo la instalación en un espacio de nombres diferente:

```
operator-sdk run bundle \  
$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/dynamic-sizing-k8s-operator-olm-bundle/  
emr-$RELEASE-dynamic-sizing-k8s-operator:latest \  
-n operator-namespace
```

Note

Si el espacio de nombres que especifique no existe, OLM no instalará el operador. Para obtener más información, consulte [No se encontró el espacio de nombres de Kubernetes](#).

4. Compruebe que haya instalado correctamente el operador con la herramienta de línea de comandos `kubectl` de Kubernetes.

```
kubectl get csv -n operator-namespace
```

El comando `kubectl` debe devolver su operador de escalado vertical recién implementado con un estado de Fase Correcto. Si tiene problemas con la instalación o la configuración, consulte [Solución de problemas en el escalado automático vertical de Amazon EMR en EKS](#).

Cómo empezar a utilizar el escalado automático vertical de Amazon EMR en EKS

Utilice el escalado automático vertical en Amazon EMR en EKS cuando desee un ajuste automático de los recursos de memoria y CPU para adaptarlos a la carga de trabajo de la aplicación de Spark de Amazon EMR. Para obtener más información, consulte [Uso del escalado automático vertical con trabajos de Spark de Amazon EMR](#).

Envío de un trabajo de Spark con escalado automático vertical

Cuando envíes un trabajo a través de la [StartJobRunAPI](#), añada las dos configuraciones siguientes al controlador para que tu trabajo de Spark active el escalado automático vertical:

```
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/  
dynamic.sizing":"true",  
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/  
dynamic.sizing.signature":"YOUR_JOB_SIGNATURE"
```

En el código anterior, la primera línea habilita la capacidad de escalado automático vertical. La siguiente línea es una configuración de firma obligatoria que le permite elegir una firma para su trabajo.

Para obtener más información sobre estas configuraciones y los valores de parámetros aceptables, consulte [Configuración del escalado automático vertical de Amazon EMR en EKS](#). De forma predeterminada, su trabajo se envía en el modo Desactivado de escalado automático vertical solo de supervisión. Este estado de supervisión le permite calcular y ver las recomendaciones de recursos sin llevar a cabo el escalado automático. Para obtener más información, consulte [Modos de escalado automático vertical](#).

En el siguiente ejemplo, se muestra cómo completar un ejemplo de comando `start-job-run` con escalado automático vertical:

```
aws emr-containers start-job-run \  
--virtual-cluster-id $VIRTUAL_CLUSTER_ID \  
--name $JOB_NAME \  
--execution-role-arn $EMR_ROLE_ARN \  
--release-label emr-6.10.0-latest \  
--job-driver '{  
  "sparkSubmitJobDriver": {  
    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py"  
  }  
' \  
--configuration-overrides '{  
  "applicationConfiguration": [{  
    "classification": "spark-defaults",  
    "properties": {  
      "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/  
dynamic.sizing": "true",  
      "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/  
dynamic.sizing.signature": "test-signature"  
    }  
  }]  
'
```


Verificación de la funcionalidad de escalado automático vertical

Para comprobar que el escalado automático vertical funcione correctamente en el trabajo enviado, use `kubectl` para obtener el recurso personalizado `verticalpodautoscaler` y ver sus recomendaciones de escalado. Por ejemplo, el siguiente comando solicita recomendaciones sobre el trabajo de ejemplo de la sección [Envío de un trabajo de Spark con escalado automático vertical](#):

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=test-signature
```

El resultado de esta consulta debe parecerse al siguiente:

NAME	MODE	CPU	MEM
PROVIDED AGE			
ds-jceyefkxnhrvdzw6djum3naf2abm6o63a6dvjkkedqtkhlrf25eq-vpa	Off	3304504865	True
87m			

Si el resultado no es similar o contiene un código de error, consulte [Solución de problemas en el escalado automático vertical de Amazon EMR en EKS](#) para ver los pasos que le ayudarán a resolver el problema.

Configuración del escalado automático vertical de Amazon EMR en EKS

Puede configurar el escalado automático vertical al enviar trabajos de Amazon EMR Spark a través [StartJobRun](#) de la API. Establezca los parámetros de configuración relacionados con el escalado automático en el pod controlador de Spark, tal como se muestra en el ejemplo de [Envío de un trabajo de Spark con escalado automático vertical](#).

El operador de escalado automático vertical de Amazon EMR en EKS escucha los pods controladores que tienen escalado automático y, a continuación, configura la integración con el escalador automático vertical de pods (VPA) de Kubernetes con la configuración del pod controlador. Esto facilita el seguimiento de los recursos y el escalado automático de los pods ejecutores de Spark.

En las siguientes secciones, se describen los parámetros que puede usar al configurar el escalado automático vertical para su clúster de Amazon EKS.

Note

Configure el parámetro de alternancia de características como una etiqueta y configure los parámetros restantes como anotaciones en el pod controlador de Spark. Los parámetros de

escalado automático pertenecen al dominio `emr-containers.amazonaws.com/` y tienen el prefijo `dynamic.sizing`.

Parámetros necesarios

Cuando envíe su trabajo, debe incluir los dos parámetros siguientes en el controlador de tareas de Spark:

Clave	Descripción	Valores aceptados	Valor predeterminado	Tipo	Parámetro de Spark ¹
<code>dynamic.sizing</code>	Conmutador de características	<code>true</code> , <code>false</code>	no configurado	etiqueta	<code>spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing</code>
<code>dynamic.sizing.signature</code>	Firma de trabajo	<code>string</code>	no configurado	anotación	<code>spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.signature</code>

¹ Utilice este parámetro como `SparkSubmitParameter` o `ConfigurationOverride` en la API de `StartJobRun`.

- **dynamic.sizing:** puede activar y desactivar el escalado automático vertical con la etiqueta `dynamic.sizing`. Para activar el escalado automático vertical, establezca `dynamic.sizing` en `true` en el pod controlador de Spark. Si omite esta etiqueta o la establece en cualquier otro valor que no sea `true`, el escalado automático vertical está desactivado.
- **dynamic.sizing.signature:** defina la firma del trabajo con la anotación `dynamic.sizing.signature` en el pod controlador. El escalado automático vertical agrega los datos de uso de recursos en diferentes ejecuciones de trabajos de Amazon EMR Spark para obtener recomendaciones de recursos. Usted proporciona el identificador único para vincular los trabajos.

Note

Si el trabajo se repite en un intervalo fijo, como una vez al día o una vez por semana, la firma del trabajo debe permanecer igual para cada nueva instancia del trabajo. Esto garantiza que el escalado automático vertical pueda calcular y agregar las recomendaciones en diferentes ejecuciones del trabajo.

¹ Utilice este parámetro como `SparkSubmitParameter` o `ConfigurationOverride` en la API de `StartJobRun`.

Parámetros opcionales

El escalado automático vertical también admite los siguientes parámetros opcionales. Configúrelos como anotaciones en el pod controlador.

Clave	Descripción	Valores aceptados	Valor predeterminado	Tipo	Parámetro de Spark ¹
<code>dynamic.sizing.mode</code>	Modo de escalado automático vertical	Off, Initial, Auto	Off	anotación	<code>spark.kubernetes.driver.annotation.emr-containers.amazonaws.com</code>

Clave	Descripción	Valores aceptados	Valor predeterminado	Tipo	Parámetro de Spark ¹
					/dynamic.sizing.mode
dynamic.sizing.scale.memory	Permite escalar la memoria	<i>true, false</i>	true	anotación	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory
dynamic.sizing.scale.cpu	Activar o desactivar escalado de CPU	<i>true, false</i>	false	anotación	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu

Clave	Descripción	Valores aceptados	Valor predeterminado	Tipo	Parámetro de Spark ¹
dynamic.sizing.scale.memory.min	Límite mínimo de escalado de memoria	cadena, cantidad de recursos de K8 , por ejemplo: 1G	no configurado	anotación	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.min
dynamic.sizing.scale.memory.max	Límite máximo de escalado de memoria	cadena, cantidad de recursos de K8 , por ejemplo: 4G	no configurado	anotación	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max

Clave	Descripción	Valores aceptados	Valor predeterminado	Tipo	Parámetro de Spark ¹
dynamic.sizing.scale.cpu.min	Límite mínimo de escalado de CPU	cadena, cantidad de recursos de K8 , por ejemplo: 1	no configurado	anotación	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min
dynamic.sizing.scale.cpu.max	Límite máximo de escalado de CPU	cadena, cantidad de recursos de K8 , por ejemplo: 2	no configurado	anotación	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.max

Modos de escalado automático vertical

El parámetro `mode` se asigna a los diferentes modos de escalado automático que el VPA admite. Utilice la anotación `dynamic.sizing.mode` del pod controlador para configurar el modo. Se admiten los siguientes valores para este parámetro:

- **Desactivado:** un modo de ejecución en seco en el que puede supervisar las recomendaciones, pero no se lleva a cabo el escalado automático. Este es el modo predeterminado para el escalado automático vertical. En este modo, el recurso del escalador automático vertical de pods asociado

calcula las recomendaciones y usted puede supervisarlas mediante herramientas como `kubectl`, Prometheus y Grafana.

- **Inicial:** en este modo, el VPA escala automáticamente los recursos cuando se inicia el trabajo si hay recomendaciones disponibles en función del historial de ejecución del trabajo, como en el caso de un trabajo periódico.
- **Automático:** en este modo, el VPA expulsa los pods ejecutores de Spark y los escala automáticamente con la configuración de recursos recomendada cuando el pod controlador de Spark los reinicia. A veces, el VPA expulsa los pods ejecutores de Spark en ejecución, por lo que podría producirse una latencia adicional al volver a intentar el ejecutor interrumpido.

Escalado de recursos

Al configurar el escalado automático vertical, puede elegir si desea escalar los recursos de CPU y memoria. Establezca las anotaciones `dynamic.sizing.scale.cpu` y `dynamic.sizing.scale.memory` en `true` o `false`. De forma predeterminada, el escalado de CPU está establecido en `false` y el escalado de memoria está establecido en `true`.

Mínimos y máximos de recursos (límites)

De forma opcional, también puede establecer límites en los recursos de CPU y memoria. Elija un valor mínimo y máximo para estos recursos con las anotaciones `dynamic.sizing.[memory/cpu].[min/max]` cuando habilite el escalado automático. De forma predeterminada, los recursos no tienen limitaciones. Configure las anotaciones como valores de cadena que representen la cantidad de recursos de Kubernetes. Por ejemplo, configure `dynamic.sizing.memory.max` en `4G` para representar 4 GB.

Supervisión del escalado automático vertical de Amazon EMR en EKS

Puede usar la herramienta de línea de comandos `kubectl` de Kubernetes para enumerar las recomendaciones activas relacionadas con el escalado automático vertical de su clúster. También puede ver las firmas de trabajo rastreadas y eliminar los recursos innecesarios que estén asociados a las firmas.

Enumerar las recomendaciones de escalado automático vertical para el clúster

Use `kubectl` para obtener el recurso `verticalpodautoscaler` y ver el estado actual y las recomendaciones. La siguiente consulta de ejemplo devuelve todos los recursos activos de su clúster de Amazon EKS.

```
kubectl get verticalpodautoscalers \
-o custom-columns="NAME:.metadata.name, "\
"SIGNATURE:.metadata.labels.emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature, "\
"MODE:.spec.updatePolicy.updateMode, "\
"MEM:.status.recommendation.containerRecommendations[0].target.memory" \
--all-namespaces
```

El resultado de esta consulta se parece al siguiente:

NAME	SIGNATURE	MODE	MEM
ds- <i>example-id-1</i> -vpa	<i>job-signature-1</i>	Off	<i>none</i>
ds- <i>example-id-2</i> -vpa	<i>job-signature-2</i>	Initial	12936384283

Consultar y eliminar las recomendaciones de escalado automático vertical para el clúster

Al eliminar un recurso de ejecución de trabajos de escalado automático vertical de Amazon EMR, se elimina automáticamente el objeto del VPA asociado que rastrea y almacena las recomendaciones.

En el siguiente ejemplo, se utiliza kubectl para purgar las recomendaciones de un trabajo que se identifica mediante una firma:

```
kubectl delete jobrun -n emr -l=emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature=integ-test
jobrun.dynamicsizing.emr.services.k8s.aws "ds-job-signature" deleted
```

Si no conoce la firma de trabajo específica o quiere purgar todos los recursos del clúster, puede usar `--all` o `--all-namespaces` en el comando en lugar del ID de trabajo único, tal como se muestra en el siguiente ejemplo:

```
kubectl delete jobruns --all --all-namespaces
jobrun.dynamicsizing.emr.services.k8s.aws "ds-example-id" deleted
```

Desinstalar el operador de escalado automático vertical de Amazon EMR en EKS

Si desea eliminar el operador de escalado automático vertical de su clúster de Amazon EKS, utilice el comando `cleanup` con la CLI de Operator SDK, tal como se muestra en el siguiente ejemplo. Esto

también elimina las dependencias ascendentes que se instalaron con el operador, como el escalador automático vertical de pods.

```
operator-sdk cleanup emr-dynamic-sizing
```

Si hay algún trabajo en ejecución en el clúster al eliminar el operador, esos trabajos seguirán ejecutándose sin la configuración de escalado automático vertical. Si envía trabajos en el clúster después de eliminar el operador, Amazon EMR en EKS ignorará cualquier parámetro relacionado con el escalado automático vertical que haya definido durante la [configuración](#).

Ejecución de cargas de trabajo interactivas en Amazon EMR en EKS

Un punto de conexión interactivo es una puerta de enlace que conecta Amazon EMR Studio con Amazon EMR en EKS para que pueda ejecutar cargas de trabajo interactivas. Puede usar puntos de conexión interactivos con EMR Studio para ejecutar análisis interactivos con conjuntos de datos en almacenes de datos como [Amazon S3](#) y [Amazon DynamoDB](#).

Casos de uso

- Cree un script de ETL con la experiencia del IDE de EMR Studio. El IDE ingiere los datos en las instalaciones y los almacena en Amazon S3 después de las transformaciones para su posterior análisis.
- Use cuadernos para explorar los conjuntos de datos y entrene un modelo de machine learning para detectar anomalías en los conjuntos de datos.
- Cree scripts que generen informes diarios para aplicaciones analíticas, como los cuadros de mando empresariales.

Temas

- [Resumen de los puntos de conexión interactivos](#)
- [Requisitos previos para crear un punto de conexión interactivo en Amazon EMR en EKS](#)
- [Creación de un punto de conexión interactivo para su clúster virtual](#)
- [Configuración de los ajustes de los puntos de conexión interactivos](#)
- [Supervisión de puntos de conexión interactivos](#)
- [Uso de cuadernos Jupyter autoalojados](#)
- [Obtener información sobre los puntos de conexión interactivos con comandos CLI](#)

Resumen de los puntos de conexión interactivos

Un punto de conexión interactivo proporciona la capacidad para que clientes interactivos como Amazon EMR Studio se conecten a clústeres de Amazon EMR en EKS para ejecutar cargas de trabajo interactivas. El punto de conexión interactivo está respaldado por una puerta de enlace de

Jupyter Enterprise que proporciona la capacidad de administración remota del ciclo de vida de los kernels que necesitan los clientes interactivos. Los kernels son procesos específicos del lenguaje que interactúan con el cliente de Amazon EMR Studio basado en Jupyter para ejecutar cargas de trabajo interactivas.

Los puntos de conexión interactivos admiten los siguientes kernels:

- Python 3
- PySpark en Kubernetes
- Apache Spark con Scala

 Note

Los precios de Amazon EMR en EKS se aplican a los kernels y puntos de conexión interactivos. Para obtener más información, consulte la página [Precios de Amazon EMR en EKS](#).

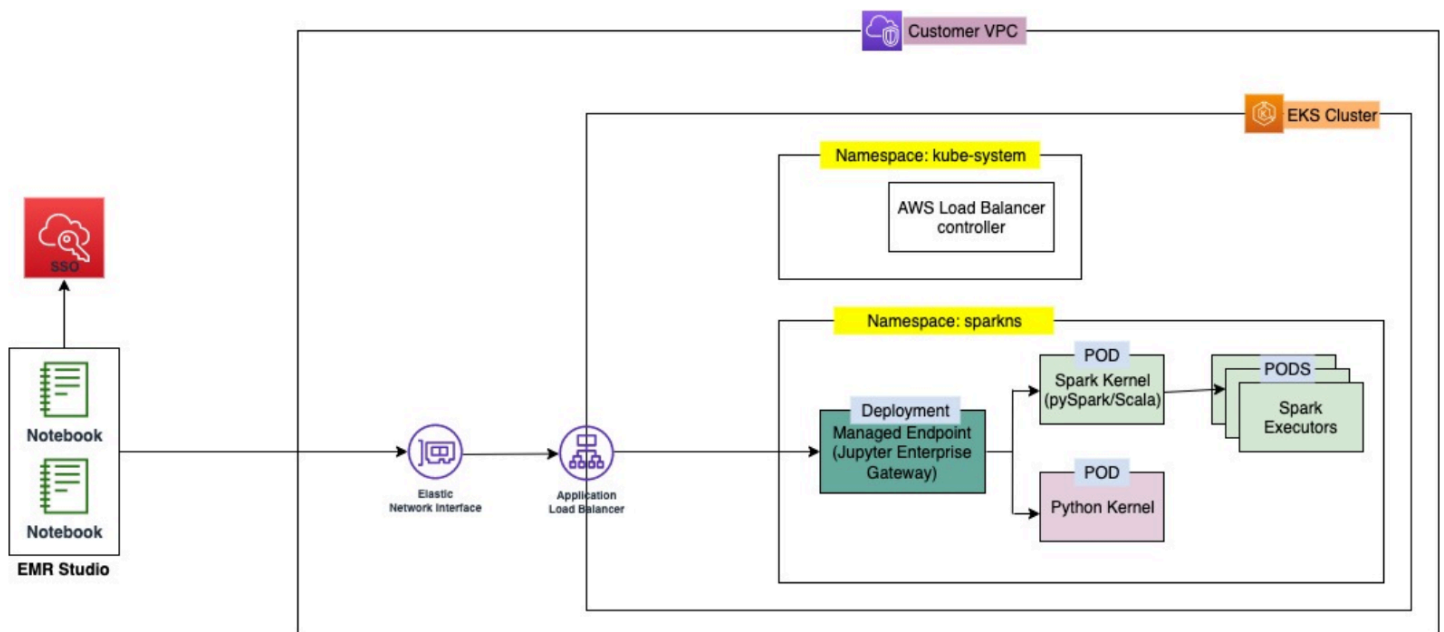
Se requieren las siguientes entidades para que EMR Studio se conecte con Amazon EMR en EKS.

- Clúster virtual en Amazon EMR en EKS: un clúster virtual es un espacio de nombres de Kubernetes con el que registra Amazon EMR. Amazon EMR utiliza clústeres virtuales para ejecutar trabajos y alojar puntos de conexión. Puede respaldar varios clústeres virtuales con el mismo clúster físico. Sin embargo, cada clúster virtual se asigna a un espacio de nombres de un clúster de Amazon EKS. Los clústeres virtuales no crean ningún recurso activo que contribuya a su factura o que requiera una administración del ciclo de vida externa al servicio.
- Punto de conexión interactivo de Amazon EMR en EKS: un punto de conexión interactivo es un punto de conexión HTTPS al que los usuarios de EMR Studio pueden conectar un espacio de trabajo. Solo puede acceder a los puntos de conexión HTTPS desde EMR Studio y crearlos en una subred privada de Amazon Virtual Private Cloud (Amazon VPC) para su clúster de Amazon EKS.

Los núcleos de Python y Spark Scala utilizan los permisos definidos en su función de ejecución de tareas de Amazon EMR en EKS para invocar otras. PySpark Servicios de AWS Todos los kernels y usuarios que se conectan al punto de conexión interactivo utilizan el rol que usted especificó al crear el punto de conexión. Le recomendamos que cree puntos de enlace independientes para los distintos usuarios y que los usuarios tengan funciones diferentes AWS Identity and Access Management (de IAM).

- **AWS Controlador Application Load Balancer:** el controlador AWS Application Load Balancer administra Elastic Load Balancing para un clúster de Amazon EKS Kubernetes. El controlador proporciona un equilibrador de carga de aplicación (ALB) al crear un recurso de Kubernetes Ingress. Un ALB expone un servicio de Kubernetes, como un punto de conexión interactivo, fuera del clúster de Amazon EKS, pero dentro de la misma Amazon VPC. Al crear un punto de conexión interactivo, también se implementa un recurso de Ingress que expone el punto de conexión interactivo mediante el ALB para que los clientes interactivos puedan conectarse a él. Solo necesita instalar un controlador AWS Application Load Balancer para cada clúster de Amazon EKS.

En el siguiente diagrama, se muestra la arquitectura de puntos de conexión interactivos de Amazon EMR en EKS. Un clúster de Amazon EKS comprende la computación para ejecutar las cargas de trabajo analíticas y el punto de conexión interactivo. El controlador de equilibrador de carga de aplicación se ejecuta en el espacio de nombres kube-system; las cargas de trabajo y los puntos de conexión interactivos se ejecutan en el espacio de nombres que especifique al crear el clúster virtual. Al crear un punto de conexión interactivo, el plano de control de Amazon EMR en EKS crea la implementación del punto de conexión interactivo en el clúster de Amazon EKS. Además, el controlador del balanceador de carga crea una instancia de entrada al balanceador de AWS carga de aplicaciones. El equilibrador de carga de aplicación proporciona la interfaz externa para que clientes como EMR Studio se conecten al clúster de Amazon EMR y ejecuten cargas de trabajo interactivas.



Requisitos previos para crear un punto de conexión interactivo en Amazon EMR en EKS

En esta sección, se describen los requisitos previos para configurar un punto de conexión interactivo que EMR Studio pueda usar para conectarse a un clúster de Amazon EMR en EKS y ejecutar cargas de trabajo interactivas.

AWS CLI

Siga los pasos que se indican en [Instalar o actualizar a la versión más reciente de](#) () AWS CLI para instalar la última versión de AWS Command Line Interface ()AWS CLI.

Instalación de eksctl

Siga los pasos que se indican en [Instalar kubectl](#) para instalar la última versión de eksctl. Si utiliza Kubernetes 1.22 o una versión posterior para su clúster de Amazon EKS, utilice una versión de eksctl posterior a 0.117.0.

Clúster de Amazon EKS

Cree un clúster de Amazon EKS. Registre el clúster como clúster virtual con Amazon EMR en EKS. A continuación, se detallan los requisitos y consideraciones para este clúster:

- El clúster debe estar en la misma Amazon Virtual Private Cloud (VPC) que EMR Studio.
- El clúster debe tener al menos una subred privada para activar los puntos de conexión interactivos, vincular los repositorios basados en Git y lanzar el equilibrador de carga de aplicación en modo privado.
- Debe haber al menos una subred privada en común entre EMR Studio y el clúster de Amazon EKS que utilice para registrar el clúster virtual. Esto garantiza que el dispositivo de punto de conexión interactivo aparezca como una opción en sus espacios de trabajo de Studio y activa la conectividad desde Studio al equilibrador de carga de aplicación.

Hay dos métodos entre los que puede elegir para conectar su Studio y su clúster de Amazon EKS:

- Cree un clúster de Amazon EKS y asócielo a las subredes que pertenecen a EMR Studio.
- Como alternativa, cree un EMR Studio y especifique las subredes privadas de su clúster de Amazon EKS.

- Los ARM optimizados para Amazon EKS y Amazon Linux no AMIs son compatibles con Amazon EMR en los puntos de enlace interactivos de EKS.
- Los puntos de conexión interactivos funcionan con los clústeres de Amazon EKS que utilizan versiones de Kubernetes anteriores a 1.30.
- Solo se admiten los [grupos de nodos administrados por Amazon EKS](#).

Conceder acceso al clúster para Amazon EMR en EKS

Siga los pasos de [Conceder acceso al clúster para Amazon EMR en EKS](#) para otorgar a Amazon EMR en EKS acceso a un espacio de nombres específico de su clúster.

Activar los IRSA en el clúster de Amazon EKS

Para activar los roles de IAM para las cuentas de servicio (IRSA) en el clúster de Amazon EKS, siga los pasos que se indican en [Habilitar roles de IAM para las cuentas de servicio \(IRSA\)](#).

Crear un rol de ejecución de trabajos de IAM

Debe crear un rol de IAM para ejecutar cargas de trabajo en puntos de conexión interactivos de Amazon EMR. En esta documentación, nos referimos a este rol de IAM como rol de ejecución de trabajos. Este rol de IAM se asigna tanto al contenedor de punto de conexión interactivo como a los contenedores de ejecución reales que se crean al enviar trabajos con EMR Studio. El nombre de recurso de Amazon (ARN) de su rol de ejecución de trabajos de Amazon EMR en EKS. Hay dos pasos necesarios para esto:

- [Cree un rol de IAM de ejecución de trabajos](#).
- [Actualice la política de confianza del rol de ejecución de trabajos](#).

Otorgar a los usuarios acceso a Amazon EMR en EKS

La entidad de IAM (usuario o rol) que realiza la solicitud para crear un punto final interactivo también debe tener los siguientes `emr-containers` permisos de Amazon EC2. Siga los pasos descritos en [Otorgar a los usuarios acceso a Amazon EMR en EKS](#) para otorgar estos permisos que permiten a Amazon EMR en EKS crear, administrar y eliminar los grupos de seguridad que limitan el tráfico entrante al equilibrador de carga de su punto de conexión interactivo.

Los siguientes permisos `emr-containers` permiten al usuario llevar a cabo operaciones básicas en puntos de conexión interactivos:

```
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress"

"emr-containers:CreateManagedEndpoint",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers>DeleteManagedEndpoint"
```

Registrar el clúster de Amazon EKS con Amazon EMR

Configure un clúster virtual y asígnelo al espacio de nombres del clúster de Amazon EKS en el que desee ejecutar sus trabajos. Para los clústeres AWS Fargate exclusivos, utilice el mismo espacio de nombres tanto para el clúster virtual Amazon EMR on EKS como para el perfil de Fargate.

Para obtener información sobre la configuración de un clúster virtual de Amazon EMR en EKS, consulte [Registrar el clúster de Amazon EKS con Amazon EMR](#).

Implemente el controlador AWS Load Balancer en el clúster de Amazon EKS

Se necesita un AWS Application Load Balancer para su clúster de Amazon EKS. Solo tiene que configurar un controlador de equilibrador de carga de aplicación por clúster de Amazon EKS. Para obtener información sobre la configuración del controlador Load Balancer de AWS aplicaciones, consulte [Instalación del complemento Load AWS Balancer Controller en la](#) Guía del usuario de Amazon EKS.

Creación de un punto de conexión interactivo para su clúster virtual

Este tema describe un par de formas de crear un punto final interactivo mediante la interfaz de línea de AWS comandos (AWS CLI) e incluye detalles sobre los parámetros de configuración disponibles.

Crear un punto de conexión interactivo con el comando **create-managed-endpoint**

Especifique los parámetros en el comando `create-managed-endpoint` de la siguiente manera. Amazon EMR en EKS admite la creación de puntos de conexión interactivos con las versiones 6.7.0 y posteriores de Amazon EMR.

```
aws emr-containers create-managed-endpoint \
--type JUPYTER_ENTERPRISE_GATEWAY \
--virtual-cluster-id 1234567890abcdef0xxxxxxx \
--name example-endpoint-name \
--execution-role-arn arn:aws:iam::444455556666:role/JobExecutionRole \
--release-label emr-6.9.0-latest \
--configuration-overrides '{
  "applicationConfiguration": [{
    "classification": "spark-defaults",
    "properties": {
      "spark.driver.memory": "2G"
    }
  }],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "log_group_name",
      "logStreamNamePrefix": "log_stream_prefix"
    },
    "persistentAppUI": "ENABLED",
    "s3MonitoringConfiguration": {
      "logUri": "s3://my_s3_log_location"
    }
  }
}'
```

Para obtener más información, consulte [Parámetros para crear un punto de conexión interactivo](#).

Crear un punto de conexión interactivo con parámetros especificados en un archivo JSON

1. Cree un archivo `create-managed-endpoint-request.json` y especifique los parámetros necesarios para su punto de conexión, tal como se muestra en el siguiente archivo JSON:

```
{
```



```

"name": "MY_TEST_ENDPOINT",
"virtualClusterId": "MY_CLUSTER_ID",
"type": "JUPYTER_ENTERPRISE_GATEWAY",
"releaseLabel": "emr-6.9.0-latest",
"executionRoleArn": "arn:aws:iam::444455556666:role/JobExecutionRole",
"configurationOverrides":
{
  "applicationConfiguration":
  [
    {
      "classification": "spark-defaults",
      "properties":
      {
        "spark.driver.memory": "8G"
      }
    }
  ],
  "monitoringConfiguration":
  {
    "persistentAppUI": "ENABLED",
    "cloudWatchMonitoringConfiguration":
    {
      "logGroupName": "my_log_group",
      "logStreamNamePrefix": "log_stream_prefix"
    },
    "s3MonitoringConfiguration":
    {
      "logUri": "s3://my_s3_log_location"
    }
  }
}
}

```

2. Utilice el comando `create-managed-endpoint` con una ruta al archivo `create-managed-endpoint-request.json` que esté almacenado localmente o en Amazon S3.

```

aws emr-containers create-managed-endpoint \
--cli-input-json file:///./create-managed-endpoint-request.json --region AWS-Region

```

Resultado de crear un punto de conexión interactivo

Debería ver el siguiente resultado en el terminal. El resultado incluye el nombre y el identificador del nuevo punto de conexión interactivo:

```
{
  "id": "1234567890abcdef0",
  "name": "example-endpoint-name",
  "arn": "arn:aws:emr-containers:us-west-2:111122223333:/
virtualclusters/444455556666/endpoints/444455556666",
  "virtualClusterId": "111122223333xxxxxxxx"
}
```

La ejecución de `aws emr-containers create-managed-endpoint` crea un certificado autofirmado que permite la comunicación HTTPS entre EMR Studio y el servidor del punto de conexión interactivo.

Si ejecuta `create-managed-endpoint` y no ha completado los requisitos previos, Amazon EMR devuelve un mensaje de error con las acciones que debe llevar a cabo para continuar.

Parámetros para crear un punto de conexión interactivo

Temas

- [Parámetros obligatorios para los puntos de conexión interactivos](#)
- [Parámetros opcionales para los puntos de conexión interactivos](#)

Parámetros obligatorios para los puntos de conexión interactivos

Debe especificar los siguientes parámetros cuando cree un punto de conexión interactivo:

--type

Utilice `JUPYTER_ENTERPRISE_GATEWAY`. Este es el único tipo admitido.

--virtual-cluster-id

El identificador del clúster virtual que registró con Amazon EMR en EKS.

--name

Un nombre descriptivo para el punto de conexión interactivo que ayuda a los usuarios de EMR Studio a seleccionarlo de la lista desplegable.

--execution-role-arn

El nombre de recurso de Amazon (ARN) de su rol de ejecución de trabajos de IAM para Amazon EMR en EKS que se creó como parte de los requisitos previos.

--release-label

La etiqueta de versión de la versión de Amazon EMR que se utilizará en el punto de conexión. Por ejemplo, `emr-6.9.0-latest`. Amazon EMR en EKS admite puntos de conexión interactivos con las versiones 6.7.0 y posteriores de Amazon EMR.

Parámetros opcionales para los puntos de conexión interactivos

De forma opcional, también puede especificar los parámetros siguientes al crear un punto de conexión interactivo:

--configuration-overrides

Para anular las configuraciones predeterminadas de las aplicaciones, proporcione un objeto de configuración. 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. Las clasificaciones de configuración disponibles varían en función de la versión de Amazon EMR en EKS. Para ver una lista de las clasificaciones de configuración que están disponibles para cada versión de lanzamiento de Amazon EMR en EKS, consulte [Versiones de Amazon EMR en EKS](#). Además de las clasificaciones de configuración enumeradas para cada versión, los puntos de conexión interactivos incluyen la clasificación adicional `jeg-config`. Para obtener más información, consulte [Opciones de configuración de Jupyter Enterprise Gateway \(JEG\)](#).

Configuración de los ajustes de los puntos de conexión interactivos

Esta sección contiene una serie de temas que abarcan diversas configuraciones de puntos de conexión interactivos y configuraciones de pods. Esto le permite supervisar y solucionar errores, enviar información de registro a Amazon S3 o a Amazon S3 o crear puntos de enlace interactivos en los que especifique plantillas de pods personalizadas. Amazon CloudWatch Logs

Temas

- [Supervisión de trabajos de Spark de](#)
- [Especificar plantillas de pods personalizadas con puntos de conexión interactivos](#)
- [Implementación de un pod de JEG en un grupo de nodos](#)
- [Opciones de configuración de Jupyter Enterprise Gateway \(JEG\)](#)
- [Modificación de los parámetros PySpark de sesión](#)
- [Imagen de kernel personalizada con punto de conexión interactivo](#)

Supervisión de trabajos de Spark de

Para poder supervisar y solucionar los errores, configure los puntos de enlace interactivos de modo que los trabajos iniciados con el punto de conexión puedan enviar la información de registro a Amazon S3, Amazon CloudWatch Logs o a ambos. En las siguientes secciones, se describe cómo enviar los registros de las aplicaciones de Spark a Amazon S3 para los trabajos de Spark que lance con Amazon EMR en los puntos de conexión interactivos de EKS.

Configurar la política de IAM para los registros de Amazon S3

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

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

```
}
```

Note

Amazon EMR en EKS también puede crear un bucket de S3. Si no hay ningún bucket de S3 disponible, incluya el permiso `s3:CreateBucket` en la política de IAM.

Una vez que haya otorgado a su rol de ejecución los permisos necesarios para enviar registros al bucket de S3, los datos de registro se envían a las siguientes ubicaciones de Amazon S3. Esto sucede cuando `s3MonitoringConfiguration` se pasa a la sección `monitoringConfiguration` de una solicitud `create-managed-endpoint`.

- Registros del controlador: `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/spark-application-id-driver/(stderr.gz/stdout.gz)`
- Registros del ejecutor: `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/executor-pod-name-exec-<Number>/(stderr.gz/stdout.gz)`

Note

Amazon EMR en EKS no carga los registros de los puntos de conexión a su bucket de S3.

Especificar plantillas de pods personalizadas con puntos de conexión interactivos

Puede crear puntos de conexión interactivos en los que especifique plantillas de pods personalizadas para los controladores y ejecutores. Las plantillas de pods son especificaciones que determinan cómo ejecutar cada pod. Puede usar archivos de plantillas de pods para definir las configuraciones de los pods controladores o ejecutores que las configuraciones de Spark no admiten. Actualmente, las plantillas de pods son compatibles con las versiones 6.3.0 y posteriores de Amazon EMR.

Para obtener más información sobre las plantillas de pods, consulte [Uso de plantillas de pods](#) en la Guía de desarrollo de Amazon EMR en EKS.

En el ejemplo siguiente, se muestra cómo crear un punto de conexión interactivo con plantillas de pods:

```
aws emr-containers create-managed-endpoint \
  --type JUPYTER_ENTERPRISE_GATEWAY \
  --virtual-cluster-id virtual-cluster-id \
  --name example-endpoint-name \
  --execution-role-arn arn:aws:iam::aws-account-id:role/EKSClusterRole \
  --release-label emr-6.9.0-latest \
  --configuration-overrides '{
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.kubernetes.driver.podTemplateFile": "path/to/driver/
template.yaml",
          "spark.kubernetes.executor.podTemplateFile": "path/to/executor/
template.yaml"
        }
      }
    ]
  }'
```

Implementación de un pod de JEG en un grupo de nodos

La colocación del pod de JEG (Jupyter Enterprise Gateway) es una característica que permite implementar un punto de conexión interactivo en un grupo de nodos específico. Con esta característica, puede configurar ajustes como `instance type` para el punto de conexión interactivo.

Asociación de un pod de JEG a un grupo de nodos administrado

La siguiente propiedad de configuración le permite especificar el nombre de un grupo de nodos administrado en el clúster de Amazon EKS en el que se implementará el pod de JEG.

```
//payload
--configuration-overrides '{
  "applicationConfiguration": [
    {
      "classification": "endpoint-configuration",
      "properties": {
        "managed-nodegroup-name": NodeGroupName
      }
    }
  ]
}
```

```
    }  
  ]  
}'
```

Un grupo de nodos debe tener la etiqueta de Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` adjunta a todos los nodos que forman parte del grupo de nodos. Para enumerar todos los nodos de un grupo de nodos que tienen esta etiqueta, use el siguiente comando:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Si el resultado del comando anterior no devuelve los nodos que forman parte del grupo de nodos administrado, significa que no hay nodos en el grupo de nodos que tengan la etiqueta de Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` adjunta. En este caso, siga los pasos que se indican a continuación para adjuntar esa etiqueta a los nodos del grupo de nodos.

1. Use el comando siguiente para agregar la etiqueta de Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` a todos los nodos de un grupo de nodos administrado: *NodeGroupName*

```
kubectl label nodes --selector eks:nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

2. Compruebe que los nodos estén etiquetados correctamente con el siguiente comando:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Un grupo de nodos administrado debe estar asociado al grupo de seguridad de un clúster de Amazon EKS, lo que suele ocurrir si ha creado el clúster y el grupo de nodos administrado mediante `eksctl`. Puede verificarlo en la AWS consola mediante los siguientes pasos.

1. Vaya a su clúster en la consola de Amazon EKS.
2. Vaya a la pestaña de redes del clúster y anote su grupo de seguridad.
3. Vaya a la pestaña de procesamiento del clúster y haga clic en el nombre del grupo de nodos administrado.

4. En la pestaña Detalles del grupo de nodos administrado, compruebe que el grupo de seguridad del clúster que indicó anteriormente aparezca en Grupos de seguridad.

Si el grupo de nodos administrado no está asociado al grupo de seguridad del clúster de Amazon EKS, debe adjuntar la etiqueta `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName` al grupo de seguridad del grupo de nodos. Siga estos pasos para adjuntar la etiqueta.

1. Ve a la EC2 consola de Amazon y haz clic en los grupos de seguridad en el panel de navegación izquierdo.
2. Para seleccionar el grupo de seguridad del grupo de nodos administrado, haga clic en la casilla de verificación.
3. En la pestaña Etiquetas, agregue la etiqueta `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName` con el botón Administrar etiquetas.

Asociación de un pod de JEG a un grupo de nodos autoadministrado

La siguiente propiedad de configuración le permite especificar el nombre de un grupo de nodos autoadministrado o no administrado en el clúster de Amazon EKS en el que se implementará el pod de JEG.

```
//payload
--configuration-overrides '{
  "applicationConfiguration": [
    {
      "classification": "endpoint-configuration",
      "properties": {
        "self-managed-nodegroup-name": NodeGroupName
      }
    }
  ]
}'
```

El grupo de nodos debe tener la etiqueta de Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` adjunta a todos los nodos que forman parte del grupo de nodos. Para enumerar todos los nodos de un grupo de nodos que tienen esta etiqueta, use el comando siguiente:


```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Si el resultado del comando anterior no devuelve los nodos que forman parte del grupo de nodos autoadministrado, significa que no hay nodos en el grupo de nodos que tengan la etiqueta de Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` adjunta. En este caso, siga los pasos que se indican a continuación para adjuntar esa etiqueta a los nodos del grupo de nodos.

1. Si ha creado el grupo de nodos autoadministrado con `eksctl`, utilice el siguiente comando para agregar la etiqueta de Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` a todos los nodos del grupo de nodos autoadministrado *NodeGroupName* a la vez.

```
kubectl label nodes --selector alpha.eksctl.io/nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Si no utilizó `eksctl` para crear el grupo de nodos autoadministrado, tendrá que sustituir el selector del comando anterior por una etiqueta de Kubernetes diferente que esté adjunta a todos los nodos del grupo de nodos.

2. Utilice el siguiente comando para verificar que los nodos estén etiquetados correctamente:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

El grupo de seguridad del grupo de nodos autoadministrado debe tener la etiqueta `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName` adjunta. Siga estos pasos para adjuntar la etiqueta al grupo de seguridad desde la AWS Management Console.

1. Ve a la EC2 consola de Amazon. En el panel de navegación izquierdo, elija Grupos de seguridad.
2. Seleccione la casilla de verificación junto al grupo de seguridad de su grupo de nodos autoadministrado.
3. En la pestaña Etiquetas, use el botón Administrar etiquetas para agregar la etiqueta `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName`. Sustituya *ClusterName* y *NodeGroupName* por los valores adecuados.

Asociación de un pod de JEG a un grupo de nodos administrado con instancias bajo demanda

También puede definir etiquetas adicionales, conocidas como selectores de etiquetas de Kubernetes, para especificar restricciones o restricciones adicionales para ejecutar un punto de conexión interactivo en un nodo o grupo de nodos determinado. En el siguiente ejemplo, se muestra cómo utilizar las EC2 instancias de Amazon bajo demanda para un pod JEG.

```
--configuration-overrides '{
  "applicationConfiguration": [
    {
      "classification": "endpoint-configuration",
      "properties": {
        "managed-nodegroup-name": NodeGroupName,
        "node-labels": "eks.amazonaws.com/capacityType:ON_DEMAND"
      }
    }
  ]
}'
```

Note

Solo puede usar la propiedad `node-labels` con una propiedad `managed-nodegroup-name` o `self-managed-nodegroup-name`.

Opciones de configuración de Jupyter Enterprise Gateway (JEG)

Amazon EMR en EKS utiliza Jupyter Enterprise Gateway (JEG) para activar los puntos de conexión interactivos. Puede establecer los siguientes valores para las configuraciones de JEG permitidas cuando crea el punto de conexión.

- **`RemoteMappingKernelManager.cull_idle_timeout`**: tiempo de espera en segundos (número entero), tras el cual el kernel se considera inactivo y listo para eliminarse. Los valores iguales a 0 o inferiores desactivan la eliminación selectiva. Los tiempos de espera cortos pueden provocar que se eliminen los kernels de los usuarios con conexiones de red deficientes.
- **`RemoteMappingKernelManager.cull_interval`**: el intervalo en segundos (número entero) para comprobar si hay kernels inactivos que superen el valor de tiempo de espera de eliminación.

Modificación de los parámetros PySpark de sesión

A partir de Amazon EMR en la versión 6.9.0 de EKS, en Amazon EMR Studio puede ajustar la configuración de Spark asociada a una PySpark sesión ejecutando el `%%configure` comando mágico en la celda del portátil EMR.

En el siguiente ejemplo, se muestra una carga útil de muestra que puede usar para modificar la memoria, los núcleos y otras propiedades del controlador y ejecutor de Spark. En cuanto a los ajustes de `conf`, puede configurar cualquier configuración de Spark que se mencione en la [documentación de configuración de Apache Spark](#).

```
%%configure -f
{
  "driverMemory": "16G",
  "driverCores" 4,
  "executorMemory" : "32G"
  "executorCores": 2,
  "conf": {
    "spark.dynamicAllocation.maxExecutors" : 10,
    "spark.dynamicAllocation.minExecutors": 1
  }
}
```

En el siguiente ejemplo, se muestra una carga útil de muestra que puede usar para agregar archivos, `pyFiles` y dependencias `jar` a un tiempo de ejecución de Spark.

```
%%configure -f
{
  "files": "s3://amzn-s3-demo-bucket-emr-eks/sample_file.txt",
  "pyFiles": : "path-to-python-files",
  "jars" : "path-to-jars"
}
```

Imagen de kernel personalizada con punto de conexión interactivo

Para asegurarse de que disponga de las dependencias correctas para su aplicación cuando ejecute cargas de trabajo interactivas desde Amazon EMR Studio, puede personalizar las imágenes de Docker para los puntos de conexión interactivos y ejecutar imágenes de kernel base personalizadas. Para crear un punto de conexión interactivo y conectarlo a una imagen de Docker personalizada, siga estos pasos.

Note

Solo puede anular las imágenes base. No puede agregar nuevos tipos de imágenes de kernel.

1. Cree y publique una imagen de Docker personalizada. La imagen base contiene el tiempo de ejecución de Spark y los kernels del cuaderno que se ejecutan con él. Para crear la imagen, puede seguir los pasos del 1 al 4 de [Cómo personalizar las imágenes de Docker](#). En el paso 1, notebook-spark debe usar el URI de imagen base del archivo de Docker en lugar de spark.

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

Para obtener más información sobre cómo seleccionar Regiones de AWS y almacenar etiquetas de imagen, consulte [Información sobre cómo seleccionar un URI de imagen base](#)

2. Cree un punto de conexión interactivo que se pueda utilizar con la imagen personalizada.
 - a. Cree un archivo JSON denominado `custom-image-managed-endpoint.json` con el siguiente contenido. En este ejemplo, se utiliza la versión 6.9.0 de Amazon EMR.

Example

```
{
  "name": "endpoint-name",
  "virtualClusterId": "virtual-cluster-id",
  "type": "JUPYTER_ENTERPRISE_GATEWAY",
  "releaseLabel": "emr-6.9.0-latest",
  "executionRoleArn": "execution-role-arn",
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "jupyter-kernel-overrides",
        "configurations": [
          {
            "classification": "python3",
            "properties": {
              "container-image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/custom-notebook-python:latest"
            }
          }
        ]
      }
    ]
  }
}
```

```

    },
    {
      "classification": "spark-python-kubernetes",
      "properties": {
        "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
      }
    }
  ]
}

```

- b. Cree un punto de conexión interactivo con las configuraciones especificadas en el archivo JSON, tal como se muestra en el ejemplo siguiente. Para obtener más información, consulte [Crear un punto de conexión interactivo con el comando `create-managed-endpoint`](#).

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

3. Conéctese al punto de conexión interactivo a través de EMR Studio. Para obtener más información y los pasos a seguir, consulte [Conectarse desde Studio](#) en la sección Amazon EMR en EKS de los documentos de AWS Workshop Studio.

Supervisión de puntos de conexión interactivos

Con Amazon EMR en la versión 6.10 y posteriores de EKS, los puntos de enlace interactivos emiten métricas de CloudWatch Amazon para supervisar y solucionar problemas de las operaciones del ciclo de vida del kernel. Las métricas las activan clientes interactivos, como EMR Studio o los cuadernos de Jupyter autoalojados. Cada una de las operaciones compatibles con los puntos de conexión interactivos tiene métricas asociadas. Las operaciones se modelan como dimensiones de cada métrica, como se muestra en la siguiente tabla. Las métricas emitidas por los puntos de enlace interactivos están visibles en un espacio de nombres personalizado, en su cuenta. EMRContainers

Métrica	Descripción	Unidad
RequestCount	Número acumulado de solicitudes de una operación	Recuento

Métrica	Descripción	Unidad
	procesadas por el punto de conexión interactivo.	
RequestLatency	El tiempo desde que llegó una solicitud al punto de conexión interactivo y el punto de conexión interactivo envió una respuesta.	Milisegundos
4. XXError	Se emite cuando una solicitud de operación genera un error 4xx durante el procesamiento.	Recuento
5XXError	Se emite cuando la solicitud de una operación produce un error 5Xxx en el servidor.	Recuento
KernelLaunchSuccess	Aplicable únicamente a la CreateKernel operación. Indica el número acumulado de lanzamientos del kernel que tuvieron éxito hasta esta solicitud incluida.	Recuento
KernelLaunchFailure	Aplicable únicamente a la CreateKernel operación. Indica el número acumulado de errores de lanzamiento del kernel hasta esta solicitud incluida.	Recuento

Cada métrica de punto de conexión interactivo tiene las siguientes dimensiones asociadas:

- **ManagedEndpointId**: identificador del punto de conexión interactivo
- **OperationName**: la operación desencadenada por el cliente interactivo

Los posibles valores de la dimensión **OperationName** se muestran en la siguiente tabla:

operationName	Descripción de la operación
CreateKernel	Solicite que el punto de conexión interactivo inicie un kernel.
ListKernels	Solicite que el punto de conexión interactivo enumere los kernels que se han iniciado anteriormente con el mismo token de sesión.
GetKernel	Solicite que el punto de conexión interactivo obtenga detalles sobre un kernel específico que se haya iniciado anteriormente.
ConnectKernel	Solicite que el punto de conexión interactivo establezca la conectividad entre el cliente del cuaderno y el kernel.
ConfigureKernel	Publique <code>%%configure magic request</code> en un kernel de PySpark.
ListKernelSpecs	Solicite que el punto de conexión interactivo enumere las especificaciones del kernel disponibles.
GetKernelSpec	Solicite al punto de conexión que obtenga las especificaciones del kernel de un kernel que se haya lanzado anteriormente.
GetKernelSpecResource	Solicite que el punto de conexión interactivo obtenga recursos específicos asociados a las especificaciones del kernel que se hayan lanzado anteriormente.

Ejemplos

Para acceder a la cantidad total de kernels lanzados para un punto de conexión interactivo en un día determinado:

1. Seleccione el espacio de nombres personalizado: EMRContainers
2. Seleccione su ManagedEndpointId, OperationName – CreateKernel
3. La métrica RequestCount con la estadística SUM y el periodo 1 day proporcionará todas las solicitudes de lanzamiento del kernel hechas en las últimas 24 horas.
4. KernelLaunchSuccess La métrica con la estadística SUM y el periodo 1 day proporcionará todas las solicitudes de lanzamiento del núcleo realizadas correctamente en las últimas 24 horas.

Para acceder a la cantidad de errores del kernel para un punto de conexión interactivo en un día determinado:

1. Seleccione el espacio de nombres personalizado: EMRContainers
2. Seleccione su ManagedEndpointId, OperationName – CreateKernel
3. La métrica KernelLaunchFailure con la estadística SUM y el periodo 1 day proporcionará todas las solicitudes de lanzamiento del kernel con errores hechas en las últimas 24 horas.
También puede seleccionar las métricas 4XXError y 5XXError para saber qué tipo de error se ha producido en el lanzamiento del kernel.

Uso de cuadernos Jupyter autoalojados

Puede alojar y gestionar Jupyter o JupyterLab notebooks en una EC2 instancia de Amazon o en su propio clúster de Amazon EKS como un bloc de notas Jupyter autohospedado. A continuación, puede ejecutar cargas de trabajo interactivas con sus cuadernos de Jupyter autoalojados. En las siguientes secciones se explica el proceso de configuración e implementación de un cuaderno de Jupyter autoalojado en un clúster de Amazon EKS.

Creación de un cuaderno de Jupyter autoalojado en un clúster de EKS

- [Creación de un grupo de seguridad](#)
- [Crear un punto de conexión interactivo de Amazon EMR en EKS](#)
- [Recuperar la URL del servidor de puerta de enlace de su punto de conexión interactivo](#)

- [Recuperar un token de autenticación para conectarse al punto de conexión interactivo](#)
- [Ejemplo: implementar un bloc de notas JupyterLab](#)
- [Eliminar un cuaderno de Jupyter autoalojado](#)

Creación de un grupo de seguridad

Antes de poder crear un punto de conexión interactivo y ejecutar un Jupyter o un JupyterLab bloc de notas autohospedado, debe crear un grupo de seguridad para controlar el tráfico entre el portátil y el punto de conexión interactivo. Para usar la EC2 consola de Amazon o el EC2 SDK de Amazon para crear el grupo de seguridad, consulta los pasos de [Crear un grupo de seguridad](#) en la Guía del EC2 usuario de Amazon. Debe crear el grupo de seguridad en la VPC en la que desee implementar el servidor de su cuaderno.

Para seguir el ejemplo de esta guía, utilice la misma VPC que su clúster de Amazon EKS. Si desea alojar su portátil en una VPC diferente de la VPC de su clúster de Amazon EKS, es posible que necesite crear una conexión de emparejamiento entre ambas. VPCs Para ver los pasos para crear una conexión de emparejamiento entre dos VPCs, consulte [Crear una conexión de emparejamiento de VPC en la Guía](#) de introducción de Amazon VPC.

En el siguiente paso, necesitará el ID del grupo de seguridad para [crear un punto de conexión interactivo de Amazon EMR en EKS](#).

Crear un punto de conexión interactivo de Amazon EMR en EKS

Después de crear el grupo de seguridad de su cuaderno, siga los pasos que se indican en [Creación de un punto de conexión interactivo para su clúster virtual](#) para crear un punto de conexión interactivo. Debe proporcionar el ID del grupo de seguridad que creó para su cuaderno en [Creación de un grupo de seguridad](#).

Inserte el ID de seguridad en lugar de los siguientes ajustes de *your-notebook-security-group-id* anulación de la configuración:

```
--configuration-overrides '{
  "applicationConfiguration": [
    {
      "classification": "endpoint-configuration",
      "properties": {
        "notebook-security-group-id": "your-notebook-security-group-id"
      }
    }
  ]
}
```

```
    }  
  ],  
  "monitoringConfiguration": {  
    ...'
```

Recuperar la URL del servidor de puerta de enlace de su punto de conexión interactivo

Tras crear un punto de conexión interactivo, recupere la URL del servidor de puerta de enlace con el comando `describe-managed-endpoint` incluido en la AWS CLI. Necesita esta URL para conectar su cuaderno al punto de conexión. La URL del servidor de puerta de enlace es un punto de conexión privado.

```
aws emr-containers describe-managed-endpoint \  
--region region \  
--virtual-cluster-id virtualClusterId \  
--id endpointId
```

Inicialmente, su punto final se encuentra en `CREATING` estado. Después de unos minutos, pasa al `ACTIVE` estado. Cuando el punto final es `ACTIVE`, está listo para usarse.

Tome nota del atributo `serverUrl` que el comando `aws emr-containers describe-managed-endpoint` devuelve desde el punto de conexión activo. Necesitará esta URL para conectar su portátil al punto final al [implementar su Jupyter o portátil autohospedado](#). JupyterLab

Recuperar un token de autenticación para conectarse al punto de conexión interactivo

Para conectarte a un punto final interactivo desde un Jupyter o un JupyterLab bloc de notas, debes generar un token de sesión con la API. `GetManagedEndpointSessionCredentials` El token actúa como prueba de autenticación para conectarse al servidor del punto de conexión interactivo.

El siguiente comando se explica con más detalle con un ejemplo de resultado.

```
aws emr-containers get-managed-endpoint-session-credentials \  
--endpoint-identifier endpointArn \  
--virtual-cluster-identifier virtualClusterArn \  
--execution-role-arn executionRoleArn \  
--credential-type "TOKEN" \  

```

```
--duration-in-seconds durationInSeconds \  
--region region
```

endpointArn

El ARN de su punto de conexión. Puede encontrar el ARN en el resultado de una llamada `describe-managed-endpoint`.

virtualClusterArn

El ARN del clúster virtual.

executionRoleArn

El ARN del rol de ejecución.

durationInSeconds

La duración en segundos durante la cual el token es válido. La duración predeterminada es de 15 minutos (900) y la máxima, de 12 horas (43200).

region

La misma región que el punto de conexión.

El resultado debería parecerse al siguiente ejemplo. Toma nota del *session-token* valor que utilizarás al [implementar tu Jupyter o portátil autohospedado](#). JupyterLab

```
{  
  "id": "credentialsId",  
  "credentials": {  
    "token": "session-token"  
  },  
  "expiresAt": "2022-07-05T17:49:38Z"  
}
```

Ejemplo: implementar un bloc de notas JupyterLab

Una vez que haya completado los pasos anteriores, puede probar este procedimiento de ejemplo para implementar un JupyterLab bloc de notas en el clúster de Amazon EKS con su punto de enlace interactivo.

1. Cree un espacio de nombres para ejecutar el servidor de cuadernos.

2. Cree un archivo localmente denominado `notebook.yaml` y con el siguiente contenido. El contenido del archivo se describe a continuación.

```
apiVersion: v1
kind: Pod
metadata:
  name: jupyter-notebook
  namespace: namespace
spec:
  containers:
  - name: minimal-notebook
    image: jupyter/all-spark-notebook:lab-3.1.4 # open source image
    ports:
    - containerPort: 8888
    command: ["start-notebook.sh"]
    args: ["--LabApp.token=''"]
    env:
    - name: JUPYTER_ENABLE_LAB
      value: "yes"
    - name: KERNEL_LAUNCH_TIMEOUT
      value: "400"
    - name: JUPYTER_GATEWAY_URL
      value: "serverUrl"
    - name: JUPYTER_GATEWAY_VALIDATE_CERT
      value: "false"
    - name: JUPYTER_GATEWAY_AUTH_TOKEN
      value: "session-token"
```

Si va a implementar el cuaderno de Jupyter en un clúster exclusivo de Fargate, etiquete el pod de Jupyter con una etiqueta de `role`, tal como se muestra en el siguiente ejemplo:

```
...
metadata:
  name: jupyter-notebook
  namespace: default
  labels:
    role: example-role-name-label
spec:
  ...
```

namespace

El espacio de nombres de Kubernetes en el que se implementa el cuaderno.

serverUrl

El atributo `serverUrl` que el comando `describe-managed-endpoint` devolvió en [Recuperar la URL del servidor de puerta de enlace de su punto de conexión interactivo](#).

session-token

El atributo `session-token` que el comando `get-managed-endpoint-session-credentials` devolvió en [Recuperar un token de autenticación para conectarse al punto de conexión interactivo](#).

KERNEL_LAUNCH_TIMEOUT

El tiempo en segundos que el punto final interactivo espera a que el núcleo llegue a `RUNNING` estado. Asegúrese de que haya tiempo suficiente para que se complete el inicio del kernel configurando el tiempo de espera de inicio del kernel en un valor adecuado (máximo 400 segundos).

KERNEL_EXTRA_SPARK_OPTS

Si lo desea, puede transferir configuraciones de Spark adicionales a los kernels de Spark. Establezca esta variable de entorno con los valores como propiedad de configuración de Spark, tal como se muestra en el siguiente ejemplo:

```
- name: KERNEL_EXTRA_SPARK_OPTS
  value: "--conf spark.driver.cores=2
         --conf spark.driver.memory=2G
         --conf spark.executor.instances=2
         --conf spark.executor.cores=2
         --conf spark.executor.memory=2G
         --conf spark.dynamicAllocation.enabled=true
         --conf spark.dynamicAllocation.shuffleTracking.enabled=true
         --conf spark.dynamicAllocation.minExecutors=1
         --conf spark.dynamicAllocation.maxExecutors=5
         --conf spark.dynamicAllocation.initialExecutors=1
         "
```

3. Implemente la especificación del pod en su clúster de Amazon EKS:

```
kubectl apply -f notebook.yaml -n namespace
```

Esto pondrá en marcha un JupyterLab portátil mínimo conectado a su terminal interactivo Amazon EMR en EKS. Espere hasta que el módulo esté RUNNING. Puede comprobar su estado con el siguiente comando:

```
kubectl get pod jupyter-notebook -n namespace
```

Cuando el pod esté listo, el comando `get pod` devuelve un resultado similar al siguiente:

NAME	READY	STATUS	RESTARTS	AGE
jupyter-notebook	1/1	Running	0	46s

4. Adjunte el grupo de seguridad del cuaderno al nodo en el que está programado el cuaderno.

a. En primer lugar, identifique el nodo en el que está programado el pod de `jupyter-notebook` con el comando `describe pod`.

```
kubectl describe pod jupyter-notebook -n namespace
```

- b. Abra la consola Amazon EKS en <https://console.aws.amazon.com/eks/home#/clusters>.
- c. Vaya a la pestaña Computación de su clúster de Amazon EKS y seleccione el nodo identificado por el comando `describe pod`. Seleccione el ID de la instancia para el nodo.
- d. En el menú Acciones, seleccione Seguridad > Cambiar grupos de seguridad para adjuntar el grupo de seguridad que creó en [Creación de un grupo de seguridad](#).
- e. Si va a implementar el módulo de Jupyter Notebook AWS Fargate, cree un módulo [SecurityGroupPolicy](#) para aplicarlo al módulo de Jupyter Notebook con la siguiente etiqueta de rol:

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: example-security-group-policy-name
  namespace: default
spec:
  podSelector:
    matchLabels:
```

```

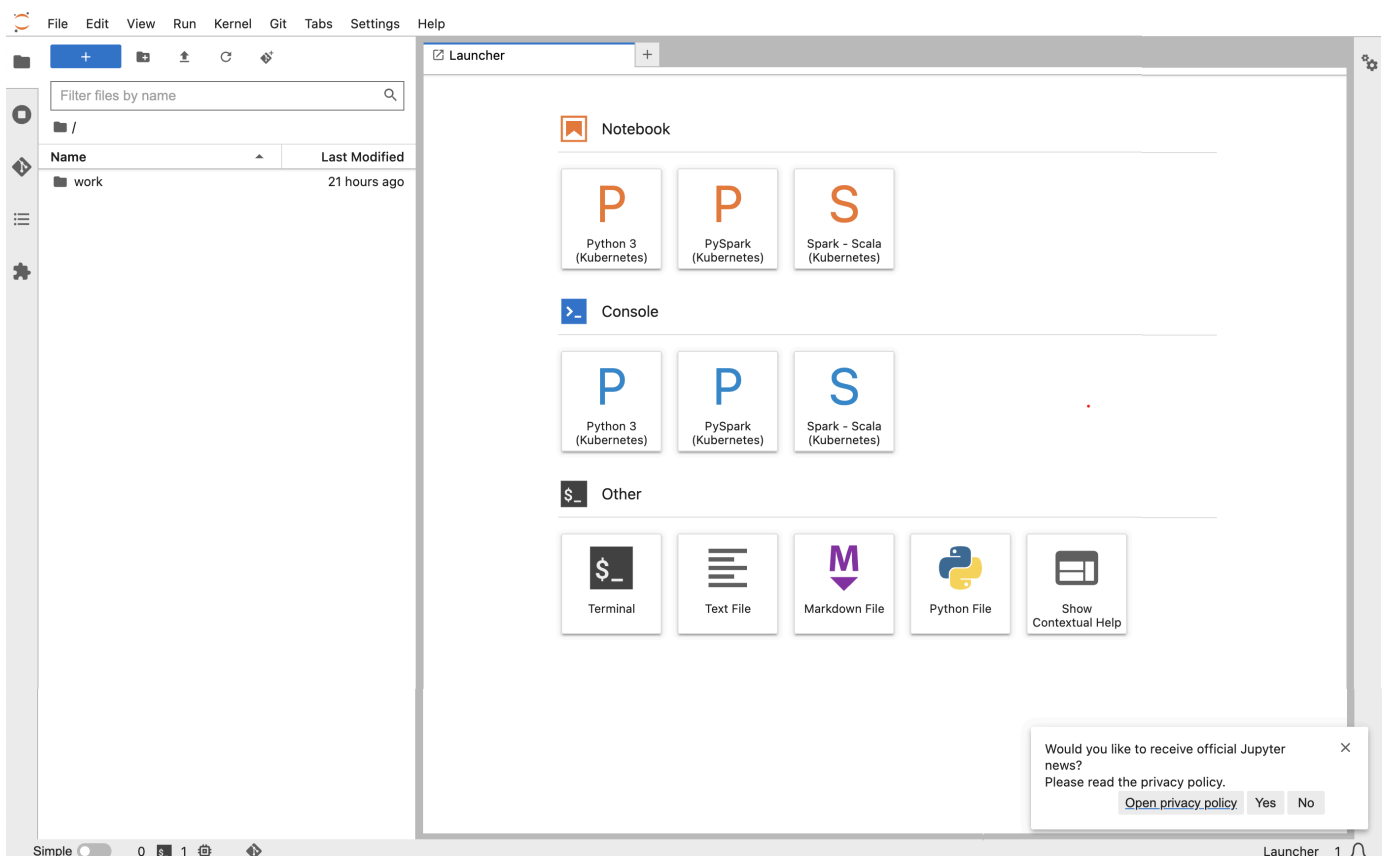
    role: example-role-name-label
securityGroups:
  groupIds:
    - your-notebook-security-group-id
EOF

```

5. Ahora, redireccione el puerto para que pueda acceder localmente a la interfaz: JupyterLab

```
kubectl port-forward jupyter-notebook 8888:8888 -n namespace
```

Una vez que se esté ejecutando, navegue hasta su navegador local y visite la interfaz `localhost:8888` para ver la JupyterLab interfaz:



6. Desde JupyterLab, crea un nuevo cuaderno de Scala. Aquí tiene un ejemplo de fragmento de código que puede ejecutar para aproximar el valor de Pi:

```

import scala.math.random
import org.apache.spark.sql.SparkSession

/** Computes an approximation to pi */
val session = SparkSession

```

```

.builder
.appName("Spark Pi")
.getOrCreate()

val slices = 2
// avoid overflow
val n = math.min(100000L * slices, Int.MaxValue).toInt

val count = session.sparkContext
.parallelize(1 until n, slices)
.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  if (x*x + y*y <= 1) 1 else 0
}.reduce(_ + _)

println(s"Pi is roughly ${4.0 * count / (n - 1)}")
session.stop()

```

The screenshot shows the Databricks IDE interface. On the left is a file explorer with a search bar and a list of files including 'work' and 'Untitled.ipynb'. The main area displays a code cell with the Scala code for approximating Pi. The code has been executed, and the output is visible below the code. The output shows the calculated value of Pi and the number of partitions used.

```

[3]: import scala.math.random
import org.apache.spark.sql.SparkSession

/** Computes an approximation to pi */
val session = SparkSession
  .builder
  .appName("Spark Pi")
  .getOrCreate()

val slices = 2
// avoid overflow
val n = math.min(100000L * slices, Int.MaxValue).toInt

val count = session.sparkContext
.parallelize(1 until n, slices)
.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  if (x*x + y*y <= 1) 1 else 0
}.reduce(_ + _)

println(s"Pi is roughly ${4.0 * count / (n - 1)}")
session.stop()

Pi is roughly 3.140955704778524
session = org.apache.spark.sql.SparkSession@722cd3ee
slices = 2
n = 200000
count = 157047

[3]: 157047

```

The bottom status bar indicates the mode is 'Command' and the current file is 'Untitled.ipynb'.

Eliminar un cuaderno de Jupyter autoalojado

Cuando lo tenga todo listo para eliminar su cuaderno autoalojado, también puede eliminar el punto de conexión interactivo y el grupo de seguridad. Lleve a cabo las acciones en el siguiente orden:

1. Utilice el siguiente comando para eliminar el pod de jupyter-notebook:

```
kubectl delete pod jupyter-notebook -n namespace
```

2. A continuación, elimine el punto de conexión interactivo con el comando `delete-managed-endpoint`. Para ver los pasos para eliminar un punto de conexión interactivo, consulte [Eliminar un punto de conexión interactivo](#). Inicialmente, su punto final estará en `TERMINATING` estado. Una vez que se han limpiado todos los recursos, pasa al `TERMINATED` estado.
3. Si no piensa utilizar el grupo de seguridad de cuadernos que creó en [Creación de un grupo de seguridad](#) para otras implementaciones de cuaderno de Jupyter, puede eliminarlo. Consulte [Eliminar un grupo de seguridad](#) en la Guía del EC2 usuario de Amazon para obtener más información.

Obtener información sobre los puntos de conexión interactivos con comandos CLI

En este tema, se tratan las operaciones admitidas en un punto de conexión interactivo que no sean [create-managed-endpoint](#).

Obtener detalles del punto de conexión interactivo

Tras crear un punto final interactivo, puede recuperar sus detalles mediante el `describe-managed-endpoint` AWS CLI comando. Inserte sus propios valores para *managed-endpoint-id*, *virtual-cluster-id*, y *region*:

```
aws emr-containers describe-managed-endpoint --id managed-endpoint-id \
--virtual-cluster-id virtual-cluster-id --region region
```

El resultado tiene un aspecto similar al siguiente, con el punto de conexión especificado, como el ARN, el ID y el nombre.

```
{
  "id": "as3ys2xxxxxxx",
```

```

"name": "endpoint-name",
  "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxxx",
  "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxxxx",
  "type": "JUPYTER_ENTERPRISE_GATEWAY",
  "state": "ACTIVE",
  "releaseLabel": "emr-6.9.0-latest",
  "executionRoleArn": "arn:aws:iam::1828xxxxxxx:role/RoleName",
  "certificateAuthority": {
    "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbbbb",
    "certificateData": "certificate-data"
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "8G"
        }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "log-group-name",
        "logStreamNamePrefix": "log-stream-name-prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3-bucket-name"
      }
    }
  },
  "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
  "createdAt": "2022-09-19T12:37:49+00:00",
  "securityGroup": "sg-aaaaaaaaaaaaaa",
  "subnetIds": [
    "subnet-1111111111",
    "subnet-2222222222",
    "subnet-3333333333"
  ],
  "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",

```

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

Enumerar todos los puntos de conexión interactivos asociados a un clúster virtual

Utilice el `list-managed-endpoints` AWS CLI comando para obtener una lista de todos los puntos finales interactivos asociados a un clúster virtual específico. Reemplace `virtual-cluster-id` por el ID del clúster virtual.

```
aws emr-containers list-managed-endpoints --virtual-cluster-id virtual-cluster-id
```

El resultado del comando `list-managed-endpoint` se muestra a continuación:

```
{
  "endpoints": [{
    "id": "as3ys2xxxxxxx",
    "name": "endpoint-name",
    "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
    "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxxxx",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "state": "ACTIVE",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "arn:aws:iam::1828xxxxxxx:role/RoleName",
    "certificateAuthority": {
      "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbbbb",
      "certificateData": "certificate-data"
    },
    "configurationOverrides": {
      "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory": "8G"
        }
      ]
    },
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "log-group-name",
        "logStreamNamePrefix": "log-stream-name-prefix"
      }
    }
  ]
}
```

```

        },
        "s3MonitoringConfiguration": {
            "logUri": "s3-bucket-name"
        }
    },
    "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
    "createdAt": "2022-09-19T12:37:49+00:00",
    "securityGroup": "sg-aaaaaaaaaaaaaa",
    "subnetIds": [
        "subnet-111111111111",
        "subnet-222222222222",
        "subnet-333333333333"
    ],
    "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
    "tags": {}
}]
}

```

Eliminar un punto de conexión interactivo

Para eliminar un punto final interactivo asociado a un Amazon EMR en un clúster virtual de EKS, utilice el `delete-managed-endpoint` AWS CLI comando. Al eliminar un punto de conexión interactivo, Amazon EMR en EKS elimina los grupos de seguridad predeterminados que se crearon para ese punto de conexión.

Especifique valores para los siguientes parámetros del comando:

- `--id`: el identificador del punto de conexión interactivo que se desea eliminar.
- `- virtual-cluster-id` - El identificador del clúster virtual asociado al punto final interactivo que desea eliminar. Este es el mismo ID de clúster virtual que se especificó cuando se creó el punto de conexión interactivo.

```
aws emr-containers delete-managed-endpoint --id managed-endpoint-id --virtual-cluster-id virtual-cluster-id
```

El comando devuelve un resultado similar al siguiente para confirmar que se haya eliminado el punto de conexión interactivo:

```
{  
  "id": "8gai4l4xxxxx",  
  "virtualClusterId": "0b0qvauoy3ch1nqodxxxxxxx"  
}
```

Carga de datos a Amazon S3 Express One Zone con Amazon EMR en EKS

A partir de las versiones 7.2.0 y posteriores de Amazon EMR, puede utilizar Amazon EMR en EKS 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

Antes de utilizar S3 Express One Zone con Amazon EMR en EKS, debe cumplir los siguientes requisitos previos:

- [Una configuración completada de Amazon EMR en EKS.](#)
- Tras configurar Amazon EMR en EKS, [cree un clúster virtual.](#)

Introducción a S3 Express One Zone

Siga estos pasos para empezar a utilizar S3 Express One Zone

1. Añada el permiso `CreateSession` a su rol de ejecución de trabajos. Cuando S3 Express One Zone realiza inicialmente una acción como `GET`, `LIST` o `PUT` en un objeto de S3, la clase de almacenamiento llama a `CreateSession` en su nombre. En el siguiente ejemplo, se muestra cómo se puede conceder el permiso `CreateSession`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "arn:aws:s3express:<AWS_REGION>:<ACCOUNT_ID>:bucket/DOC-EXAMPLE-BUCKET",
      "Action": [
        "s3express:CreateSession"
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

2. Debe usar el conector S3A de Apache Hadoop para acceder a los buckets de S3 Express, así que cambie su Amazon S3 URIs para usar el s3a esquema de uso del conector. Si no usa el esquema, puede cambiar la implementación del sistema de archivos que utiliza para los esquemas s3 y s3n.

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

```

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

```

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

```

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

```

3. En su configuración de spark-submit, utilice el proveedor de credenciales de identidad web.

```
"spark.hadoop.fs.s3a.aws.credentials.provider=com.amazonaws.auth.WebIdentityTokenCredential
```

Supervisión de trabajos

Puede usar Amazon CloudWatch Events para realizar un seguimiento de los trabajos que se ejecutan en un clúster virtual de Amazon EMR en EKS. Puede usar los eventos para hacer un seguimiento de la actividad y el estado de los trabajos que ejecute en un clúster virtual. En los temas siguientes, se muestran formas de configurar la supervisión de forma eficaz para mantener el estado de los recursos.

Temas

- [Supervisa los trabajos con Amazon CloudWatch Events](#)
- [Automatice Amazon EMR en EKS con eventos CloudWatch](#)
- [Ejemplo: configurar una regla que invoque a Lambda](#)
- [Supervise el módulo de controladores del trabajo con una política de reintentos mediante Amazon Events CloudWatch](#)

Supervisa los trabajos con Amazon CloudWatch Events

Amazon EMR en EKS emite eventos cuando el estado de una ejecución de un trabajo cambia. Cada evento proporciona información, como la fecha y hora en que ocurrió el evento, junto con más detalles, como el ID del clúster virtual y el ID de la ejecución de trabajo que se vio afectada.

Puede usar los eventos para hacer un seguimiento de la actividad y el estado de los trabajos que ejecute en un clúster virtual. También puede usar Amazon CloudWatch Events para definir una acción que se debe realizar cuando la ejecución de un trabajo genere un evento que coincida con un patrón que especifique. Los eventos son útiles para supervisar un suceso específico durante el ciclo de vida de una ejecución de trabajo. Por ejemplo, puede supervisar cuándo el estado de una ejecución de un trabajo cambia de `submitted` a `running`. Para obtener más información sobre CloudWatch los eventos, consulta la [Guía del EventBridge usuario de Amazon](#).

En la siguiente tabla, se muestran eventos de Amazon EMR en EKS, junto con el estado o cambio de estado que indica el evento, la gravedad del evento y los mensajes de eventos. Cada evento se representa como un objeto JSON que se envía automáticamente a un flujo de eventos. El objeto JSON incluye detalles adicionales sobre el evento. El objeto JSON es especialmente importante cuando se configuran reglas para el procesamiento de CloudWatch eventos mediante Events, ya que las reglas buscan hacer coincidir los patrones del objeto JSON. Para obtener más información,

consulte [los patrones de EventBridge eventos](#) de Amazon y Amazon EMR en los eventos de EKS en la Guía [EventBridge del usuario de Amazon](#).

Eventos de cambio de estado de ejecuciones de trabajo

Estado	Gravedad	Mensaje
SUBMITTED	INFO	Job Run <i>JobRunId</i> (<i>JobRunName</i>) se envió correctamente al clúster virtual <i>VirtualClusterId</i> en <i>Time</i> UTC.
RUNNING	INFO	Job Run <i>JobRunId</i> (<i>JobRunName</i>) del clúster virtual <i>VirtualClusterId</i> comenzó a ejecutarse en <i>Time</i> .
COMPLETED	INFO	Job Run <i>jobRunId</i> (<i>JobRunName</i>) en un clúster virtual <i>VirtualClusterId</i> se completó en <i>Time</i> . El Job Run comenzó a ejecutarse a las <i>Time</i> y tardó <i>Num</i> unos minutos en completarse.
CANCELLED	WARN	La solicitud de cancelación se ha realizado correctamente para Job Run <i>JobRunId</i> (<i>JobRunName</i>) en el clúster <i>VirtualClusterId</i> virtual de <i>Time</i> y la ejecución de tareas ahora está cancelada.
ERROR	ERROR	<i>VirtualClusterId</i> Error en <i>JobRunId</i> Job Run (<i>JobRunName</i>) en el clúster virtual <i>Time</i> .

Automatice Amazon EMR en EKS con eventos CloudWatch

Puede usar Amazon CloudWatch Events para automatizar sus AWS servicios y responder a eventos del sistema, como problemas de disponibilidad de las aplicaciones o cambios en los recursos. Los eventos de AWS los servicios se envían a CloudWatch Events prácticamente en tiempo real. 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. Entre las acciones que se pueden activar automáticamente se incluyen las siguientes:

- Invocar una función AWS Lambda
- Invocar el comando Amazon EC2 Run
- Desviar el evento a Amazon Kinesis Data Streams
- Activar una máquina de AWS Step Functions estados
- Notificar un tema Amazon Simple Notification Service (SNS) o una cola Amazon Simple Queue Service (SQS)

Algunos ejemplos del uso de CloudWatch Events con Amazon EMR en EKS son los siguientes:

- Activación de una función de Lambda cuando un trabajo se ejecuta correctamente
- Notificar un tema de Amazon SNS cuando se produce un error en la ejecución de un trabajo

CloudWatch Amazon EMR genera eventos para EMR Job Run State Change «detail-type:"» en EKS para SUBMITTED, RUNNINGCANCELLED, FAILED y cambios de COMPLETED estado.

Ejemplo: configurar una regla que invoque a Lambda

Siga estos pasos para configurar una regla de CloudWatch eventos que invoque a Lambda cuando se produzca un evento de «EMR Job Run State Change».

```
aws events put-rule \  
--name cwe-test \  
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

Agregue la función Lambda de su propiedad como nuevo destino y dé permiso a CloudWatch Events para invocar la función Lambda de la siguiente manera. Reemplace **123456789012** por su ID de cuenta.

```
aws events put-targets \  
--rule cwe-test \  
--targets Id=1,Arn=arn:aws:lambda:us-east-1:123456789012:function:MyFunction
```

```
aws lambda add-permission \  

```

```
--function-name MyFunction \  
--statement-id MyId \  
--action 'lambda:InvokeFunction' \  
--principal events.amazonaws.com
```

 Note

No puede escribir un programa que dependa del orden o de la existencia de eventos de notificación, ya que pueden no estar ordenados o faltar. Los eventos se emiten en la medida de lo posible.

Supervise el módulo de controladores del trabajo con una política de reintentos mediante Amazon Events CloudWatch

Mediante CloudWatch los eventos, puede supervisar los módulos de controladores que se han creado en trabajos que tienen políticas de reintentos. Para obtener más información, consulte la sección [Supervisión de un trabajo con una política de reintento](#) de esta guía.

Administración de clústeres virtuales

Un clúster virtual es un espacio de nombres de Kubernetes en el que Amazon EMR está registrado. Puede crear, describir, enumerar y eliminar clústeres virtuales. No consumen recursos adicionales en el sistema. Un único clúster virtual se asigna a un único espacio de nombres Kubernetes. Dada esta relación, puede modelar clústeres virtuales de la misma manera que modela los espacios de nombres Kubernetes para satisfacer sus necesidades. Consulte los posibles casos de uso en la documentación de [información general de conceptos de Kubernetes](#).

Para registrar Amazon EMR con un espacio de nombres de Kubernetes en un clúster de Amazon EKS, necesita el nombre del clúster de EKS y el espacio de nombres que se ha configurado para ejecutar su carga de trabajo. Estos clústeres registrados en Amazon EMR se denominan clústeres virtuales porque no administran la computación física ni el almacenamiento, sino que apuntan a un espacio de nombres de Kubernetes en el que está programada la carga de trabajo.

Note

Antes de crear un clúster virtual, debe completar los pasos del 1 al 8 que se indican en [Configuración de Amazon EMR en EKS](#).

Temas

- [Crear un clúster virtual](#)
- [Enumerar los clústeres virtuales](#)
- [Describir un clúster virtual](#)
- [Eliminar un clúster virtual](#)
- [Estados del clúster virtual](#)

Crear un clúster virtual

Ejecute el siguiente comando para crear un clúster virtual mediante el registro de Amazon EMR con un espacio de nombres en un clúster de EKS. *virtual_cluster_name* Sustitúyalo por un nombre que proporcione para el clúster virtual. *eks_cluster_name* Sustitúyalo por el nombre del clúster de EKS. Sustituya el por el espacio de nombres *namespace_name* con el que desee registrar Amazon EMR.

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

Como alternativa, puede crear un archivo JSON que incluya los parámetros necesarios para el clúster virtual, tal como se muestra en el siguiente ejemplo.

```
{
  "name": "virtual_cluster_name",
  "containerProvider": {
    "type": "EKS",
    "id": "eks_cluster_name",
    "info": {
      "eksInfo": {
        "namespace": "namespace_name"
      }
    }
  }
}
```

A continuación, ejecute el comando `create-virtual-cluster` con la ruta al archivo JSON.

```
aws emr-containers create-virtual-cluster \
--cli-input-json file:///./create-virtual-cluster-request.json
```

Note

Para validar la creación correcta de un clúster virtual, consulte el estado de los clústeres virtuales mediante la ejecución del comando `list-virtual-clusters` o en la página Clústeres virtuales de la consola de Amazon EMR.

Enumerar los clústeres virtuales

Para ver el estado de los clústeres virtuales, ejecute el siguiente comando.

```
aws emr-containers list-virtual-clusters
```

Describir un clúster virtual

Ejecute el siguiente comando para obtener más detalles sobre un clúster virtual, como el espacio de nombres, el estado y la fecha de registro. **123456**Sustitúyalo por el ID de su clúster virtual.

```
aws emr-containers describe-virtual-cluster --id 123456
```

Eliminar un clúster virtual

Ejecute el siguiente comando para eliminar un clúster virtual. **123456**Sustitúyalo por tu ID de clúster virtual.

```
aws emr-containers delete-virtual-cluster --id 123456
```

Estados del clúster virtual

En la siguiente tabla, se describen los cuatro estados posibles de un clúster virtual.

State	Descripción
RUNNING	El estado del clúster virtual es RUNNING.
TERMINATING	La terminación del clúster virtual solicitada está en curso.
TERMINATED	La terminación solicitada se ha completado.
ARRESTED	Se ha producido un error en la terminación solicitada debido a la insuficiencia de permisos.

Tutoriales de Amazon EMR en EKS

Esta sección describe casos de uso comunes para cuando trabaje con Amazon EMR en aplicaciones de EKS. Cada aplicación está especializada y para su configuración tiene sus pasos particulares. En estos temas se proporcionan instrucciones para usar cada aplicación.

Temas

- [Uso de Delta Lake con Amazon EMR en EKS](#)
- [Uso de Apache Iceberg con Amazon EMR en EKS](#)
- [Uso PyFlink](#)
- [Uso de AWS Glue con Flink](#)
- [Uso de Apache Hudi con Apache Flink](#)
- [Uso del acelerador RAPIDS para Apache Spark con Amazon EMR en EKS](#)
- [Uso de la integración de Amazon Redshift para Apache Spark en Amazon EMR en EKS](#)
- [Uso de Volcano como programador personalizado para Apache Spark en Amazon EMR en EKS](#)
- [YuniKorn Utilización como programador personalizado para Apache Spark en Amazon EMR en EKS](#)

Uso de Delta Lake con Amazon EMR en EKS

Delta Lake es un marco de almacenamiento de código abierto para crear una arquitectura de Lakehouse. A continuación, se muestra cómo configurarlo para su uso.

Para usar [Delta Lake](#) con Amazon EMR en aplicaciones de EKS

1. Cuando inicie una ejecución de trabajo para enviar una tarea de Spark en la configuración de la aplicación, incluya los archivos JAR de Delta Lake:

```
--job-driver '{"sparkSubmitJobDriver" : {  
  "sparkSubmitParameters" : "--jars local:///usr/share/aws/delta/lib/delta-  
core.jar,local:///usr/share/aws/delta/lib/delta-storage.jar,local:///usr/share/aws/  
delta/lib/delta-storage-s3-dynamodb.jar"}}'
```

Note

Las versiones 7.0.0 y posteriores de Amazon EMR utilizan Delta Lake 3.0, que cambia el nombre de `delta-core.jar` a `delta-spark.jar`. Si utiliza las versiones 7.0.0 o posteriores de Amazon EMR, asegúrese de utilizar el nombre de archivo correcto, como en el siguiente ejemplo:

```
--jars local:///usr/share/aws/delta/lib/delta-spark.jar
```

2. Incluya la configuración adicional de Delta Lake y utilice AWS Glue Data Catalog como su metaalmacén.

```
--configuration-overrides '{
  "applicationConfiguration": [
    {
      "classification" : "spark-defaults",
      "properties" : {
        "spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension",

        "spark.sql.catalog.spark_catalog":"org.apache.spark.sql.delta.catalog.DeltaCatalog",
        "spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClient"
      }
    }
  ]
}'
```

Uso de Apache Iceberg con Amazon EMR en EKS

El JAR de tiempo de ejecución de Iceberg contiene las clases de Iceberg necesarias para la compatibilidad con el tiempo de ejecución de Spark. En el siguiente procedimiento, se muestra cómo iniciar la ejecución de un trabajo mediante el tiempo de ejecución de Iceberg Spark.

Para usar Apache Iceberg con aplicaciones de Amazon EMR en EKS

1. Cuando inicie la ejecución de un trabajo para enviar un trabajo de Spark en la configuración de la aplicación, incluya el archivo JAR del tiempo de ejecución de Iceberg Spark:

```
--job-driver '{"sparkSubmitJobDriver" : {"sparkSubmitParameters" : "--jars
local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"}}'
```


2. Incluya la configuración adicional de Iceberg:

```
--configuration-overrides '{
  "applicationConfiguration": [
    "classification" : "spark-defaults",
    "properties" : {
      "spark.sql.catalog.dev.warehouse" : "s3://amzn-s3-demo-bucket/EXAMPLE-
PREFIX/ ",
      "spark.sql.extensions ":"
org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions ",
      "spark.sql.catalog.dev" : "org.apache.iceberg.spark.SparkCatalog",
      "spark.sql.catalog.dev.catalog-impl" :
"org.apache.iceberg.aws.glue.GlueCatalog",
      "spark.sql.catalog.dev.io-impl": "org.apache.iceberg.aws.s3.S3FileIO"
    }
  ]
}'
```

Para obtener más información sobre las versiones de lanzamiento de Apache Iceberg de EMR, consulte [Historial de versiones de Iceberg](#).

Configuraciones de sesión de Spark para la integración del catálogo

Configuraciones de sesión de Spark para la integración del catálogo de Iceberg AWS Glue

En este ejemplo se muestra cómo integrar Iceberg con: Rastreador de AWS Glue

```
spark-sql \
--conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.rms.type = glue \
--conf spark.sql.catalog.rms.glue.id = glue RMS catalog ID \
--conf spark.sql.catalog.rms.glue.account-id = AWS account ID \

--conf spark.sql.extensions=
org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

A continuación, se presenta una consulta de muestra:

```
SELECT * FROM rms.rmsdb.table1
```

Configuraciones de sesión de Spark para la integración del catálogo REST AWS Glue de Iceberg

En este ejemplo se muestra cómo integrar Iceberg REST con: Rastreador de AWS Glue

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

A continuación, se presenta una consulta de muestra:

```
SELECT * FROM rms.rmsdb.table1
```

Esta configuración solo funciona para Redshift Managed Storage. No se admite FGAC para Amazon S3.

Uso PyFlink

Amazon EMR es compatible con las versiones 6.15.0 y superiores de EKS. PyFlink Si ya tiene un PyFlink script, puede realizar una de las siguientes acciones:

- Cree una imagen personalizada con el PyFlink script incluido.
- Cargue su script en una ubicación de Amazon S3

Si aún no tiene un script, puede usar el siguiente ejemplo para lanzar un PyFlink trabajo. En este ejemplo se recupera el script de S3. Si utiliza una imagen personalizada con el script ya incluido en la imagen, debe actualizar la ruta del script a la ubicación donde guardó dicho script. Si el script está en una ubicación S3, Amazon EMR en EKS recuperará el script y lo colocará en el directorio `/opt/flink/usr/lib/` del contenedor Flink.

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

```
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  executionRoleArn: job-execution-role
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://S3 bucket with your script/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

Uso de AWS Glue con Flink

Amazon EMR en EKS con Apache Flink, versiones 6.15.0 y superiores, admite el uso del catálogo de datos de AWS Glue como almacén de metadatos para flujos de trabajo de SQL en streaming y por lotes.

Primero debe crear una base de datos de AWS Glue llamada `default` que sirva como catálogo SQL de Flink. Este catálogo de Flink almacena metadatos como bases de datos, tablas, particiones, vistas, funciones y otra información necesaria para acceder a los datos de otros sistemas externos.

```
aws glue create-database \
  --database-input "{\"Name\":\"default\"}"
```

Para habilitar el soporte de AWS Glue, usa una `FlinkDeployment` especificación. Esta especificación de ejemplo usa un script de Python para emitir rápidamente algunas sentencias SQL de Flink para interactuar con el catálogo de AWS Glue.

```

apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
    aws.glue.enabled: "true"
  executionRoleArn: job-execution-role-arn;
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://<S3_bucket_with_your_script/pyflink-glue-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-glue-script.py"]
    parallelism: 1
    upgradeMode: stateless

```

A continuación se muestra un ejemplo del aspecto que podría tener el PyFlink script.

```

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

def glue_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(stream_execution_environment=env)
    t_env.execute_sql("""
        CREATE CATALOG glue_catalog WITH (
            'type' = 'hive',
            'default-database' = 'default',
            'hive-conf-dir' = '/glue/confs/hive/conf',

```

```

        'hadoop-conf-dir' = '/glue/confs/hadoop/conf'
    )
        """)
t_env.execute_sql("""
    USE CATALOG glue_catalog;
        """)
t_env.execute_sql("""
    DROP DATABASE IF EXISTS eks_flink_db CASCADE;
        """)
t_env.execute_sql("""
    CREATE DATABASE IF NOT EXISTS eks_flink_db WITH ('hive.database.location-
uri'= 's3a://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/warehouse/');
        """)
t_env.execute_sql("""
    USE eks_flink_db;
        """)
t_env.execute_sql("""
    CREATE TABLE IF NOT EXISTS eks_glue_orders (
        order_number BIGINT,
        price          DECIMAL(32,2),
        buyer          ROW first_name STRING, last_name STRING,
        order_time      TIMESTAMP(3)
    ) WITH (
        'connector' = 'datagen'
    );
        """)
t_env.execute_sql("""
    CREATE TABLE IF NOT EXISTS eks_dest_glue_orders (
        order_number BIGINT,
        price          DECIMAL(32,2),
        buyer          ROW first_name STRING, last_name STRING,
        order_time      TIMESTAMP(3)
    ) WITH (
        'connector' = 'filesystem',
        'path' = 's3a://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/
warehouse/eks_dest_glue_orders',
        'format' = 'json'
    );
        """)
t_env.execute_sql("""
    CREATE TABLE IF NOT EXISTS print_table (
        order_number BIGINT,
        price          DECIMAL(32,2),
        buyer          ROW first_name STRING, last_name STRING,

```

```

        order_time    TIMESTAMP(3)
    ) WITH (
        'connector' = 'print'
    );
    """
t_env.execute_sql("""
EXECUTE STATEMENT SET
BEGIN
INSERT INTO eksdestglueorders SELECT * FROM eksglueorders LIMIT 10;
INSERT INTO print_table SELECT * FROM eksdestglueorders;
END;
""")

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

```

Uso de Apache Hudi con Apache Flink

Apache Hudi es un marco de administración de datos de código abierto con operaciones a nivel de registro, como inserción, actualización, upsert y eliminación, que puede utilizar para simplificar la administración de los datos y el desarrollo de canalizaciones de datos. Cuando se combina con una administración de datos eficiente en Amazon S3, Hudi le permite incorporar y actualizar datos en tiempo real. Hudi mantiene los metadatos de todas las operaciones que ejecuta en el conjunto de datos, de modo que todas las acciones siguen siendo atómicas y coherentes.

Apache Hudi está disponible en Amazon EMR en EKS con Apache Flink con las versiones 7.2.0 y posteriores de Amazon EMR. Consulte los pasos siguientes para aprender cómo empezar a enviar trabajos de Apache Hudi.

Envío de un trabajo de Apache Hudi

Consulte los pasos siguientes para aprender cómo enviar un trabajo de Apache Hudi.

1. Cree una base de datos de AWS Glue llamada `default`.

```
aws glue create-database --database-input '{"Name\":\"default\"}'
```

2. Siga el [Ejemplo de SQL del operador de Kubernetes de Flink](#) para crear el archivo `flink-sql-runner.jar`.

3. Cree un script SQL de Hudi como el siguiente.

```
CREATE CATALOG hudi_glue_catalog WITH (
  'type' = 'hudi',
  'mode' = 'hms',
  'table.external' = 'true',
  'default-database' = 'default',
  'hive.conf.dir' = '/glue/confs/hive/conf/',
  'catalog.path' = 's3://<hudi-example-bucket>/FLINK_HUDI/warehouse/'
);

USE CATALOG hudi_glue_catalog;
CREATE DATABASE IF NOT EXISTS hudi_db;
use hudi_db;

CREATE TABLE IF NOT EXISTS hudi-flink-example-table(
  uuid VARCHAR(20),
  name VARCHAR(10),
  age INT,
  ts TIMESTAMP(3),
  `partition` VARCHAR(20)
)
PARTITIONED BY (`partition`)
WITH (
  'connector' = 'hudi',
  'path' = 's3://<hudi-example-bucket>/hudi-flink-example-table',
  'hive_sync.enable' = 'true',
  'hive_sync.mode' = 'glue',
  'hive_sync.table' = 'hudi-flink-example-table',
  'hive_sync.db' = 'hudi_db',
  'compaction.delta_commits' = '1',
  'hive_sync.partition_fields' = 'partition',
  'hive_sync.partition_extractor_class' =
  'org.apache.hudi.hive.MultiPartKeysValueExtractor',
  'table.type' = 'COPY_ON_WRITE'
);

EXECUTE STATEMENT SET
BEGIN

INSERT INTO hudi-flink-example-table VALUES
  ('id1', 'Alex', 23, TIMESTAMP '1970-01-01 00:00:01', 'par1'),
  ('id2', 'Stephen', 33, TIMESTAMP '1970-01-01 00:00:02', 'par1'),
```

```
( 'id3', 'Julian', 53, TIMESTAMP '1970-01-01 00:00:03', 'par2' ),
( 'id4', 'Fabian', 31, TIMESTAMP '1970-01-01 00:00:04', 'par2' ),
( 'id5', 'Sophia', 18, TIMESTAMP '1970-01-01 00:00:05', 'par3' ),
( 'id6', 'Emma', 20, TIMESTAMP '1970-01-01 00:00:06', 'par3' ),
( 'id7', 'Bob', 44, TIMESTAMP '1970-01-01 00:00:07', 'par4' ),
( 'id8', 'Han', 56, TIMESTAMP '1970-01-01 00:00:08', 'par4' );
```

```
END;
```

4. Cargue su script SQL de Hudi y el archivo `flink-sql-runner.jar` en una ubicación de S3.
5. En su archivo YAML `FlinkDeployments`, establezca `hudi.enabled` en `true`.

```
spec:
  flinkConfiguration:
    hudi.enabled: "true"
```

6. Cree un archivo YAML para ejecutar su configuración. Este archivo de ejemplo se denomina `hudi-write.yaml`.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: hudi-write-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    hudi.enabled: "true"
  executionRoleArn: "<JobExecutionRole>"
  emrReleaseLabel: "emr-7.9.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/usrlib/flink-sql-runner.jar
    args: ["/opt/flink/scripts/hudi-write.sql"]
```



```
parallelism: 1
upgradeMode: stateless
podTemplate:
  spec:
    initContainers:
      - name: flink-sql-script-download
        args:
          - s3
          - cp
          - s3://<s3_location>/hudi-write.sql
          - /flink-scripts
        image: amazon/aws-cli:latest
        imagePullPolicy: Always
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
          - mountPath: /flink-scripts
            name: flink-scripts
      - name: flink-sql-runner-download
        args:
          - s3
          - cp
          - s3://<s3_location>/flink-sql-runner.jar
          - /flink-artifacts
        image: amazon/aws-cli:latest
        imagePullPolicy: Always
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
          - mountPath: /flink-artifacts
            name: flink-artifact
    containers:
      - name: flink-main-container
        volumeMounts:
          - mountPath: /opt/flink/scripts
            name: flink-scripts
          - mountPath: /opt/flink/usrlib
            name: flink-artifact
    volumes:
      - emptyDir: {}
        name: flink-scripts
      - emptyDir: {}
```

```
name: flink-artifact
```

7. Envíe un trabajo de Flink Hudi al [operador de Flink Kubernetes](#).

```
kubect1 apply -f hudi-write.yaml
```

Uso del acelerador RAPIDS para Apache Spark con Amazon EMR en EKS

Con Amazon EMR en EKS, puede ejecutar trabajos para el acelerador RAPIDS de Nvidia para Apache Spark. Este tutorial explica cómo ejecutar trabajos de Spark con RAPIDS en tipos de instancias de unidades de procesamiento EC2 gráfico (GPU). El tutorial utiliza las siguientes versiones:

- Versión de lanzamiento de Amazon EMR en EKS 6.9.0 y versiones posteriores
- Apache Spark 3.x

Puedes acelerar Spark con los tipos de instancias de EC2 GPU de Amazon mediante el complemento Nvidia [RAPIDS Accelerator para Apache Spark](#). Al utilizar estas tecnologías en conjunto, acelera los canales de ciencia de datos sin tener que hacer ningún cambio en el código. Esto reduce el tiempo de ejecución necesario para el procesamiento de datos y el entrenamiento de modelos. Al hacer más en menos tiempo, invierte menos en el costo de la infraestructura.

Antes de empezar, asegúrese de que disponga de los siguientes recursos.

- Clúster virtual de Amazon EMR en EKS
- Clúster de Amazon EKS con un grupo de nodos habilitados para GPU

Un clúster virtual de Amazon EKS es un identificador registrado para el espacio de nombres de Kubernetes en un clúster de Amazon EKS y lo administra Amazon EMR en EKS. El identificador permite a Amazon EMR utilizar el espacio de nombres de Kubernetes como destino para ejecutar trabajos. Para obtener más información sobre cómo configurar un clúster virtual, consulte [Configuración de Amazon EMR en EKS](#) en esta guía.

Debe configurar el clúster virtual de Amazon EKS con un grupo de nodos que tenga instancias de GPU. Debe configurar los nodos con un complemento de dispositivo NVIDIA. Para obtener más información, consulte [Grupos de nodos administrados](#).

Para configurar su clúster de Amazon EKS para agregar grupos de nodos habilitados para GPU, siga este procedimiento:

Para agregar grupos de nodos habilitados para GPU

1. Cree un grupo de nodos habilitado para GPU con el comando [create-nodegroup](#). Asegúrese de sustituir los parámetros correctos para su clúster de Amazon EKS. Utilice un tipo de instancia que sea compatible con RAPIDS para Spark, como P4, P3, G5 o G4dn.

```
aws eks create-nodegroup \
  --cluster-name EKS_CLUSTER_NAME \
  --nodegroup-name NODEGROUP_NAME \
  --scaling-config minSize=0,maxSize=5,desiredSize=2 CHOOSE_APPROPRIATELY \
  --ami-type AL2_x86_64_GPU \
  --node-role NODE_ROLE \
  --subnets SUBNETS_SPACE_DELIMITED \
  --remote-access ec2SshKey= SSH_KEY \
  --instance-types GPU_INSTANCE_TYPE \
  --disk-size DISK_SIZE \
  --region AWS_REGION
```

2. Instala el complemento de dispositivo de Nvidia en tu clúster para emitir el número de GPUs unidades en cada nodo del clúster y ejecutar contenedores compatibles con la GPU en tu clúster. Ejecute el siguiente comando para instalar el complemento:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/nvidia-device-plugin.yml
```

3. Para validar cuántos GPUs hay disponibles en cada nodo del clúster, ejecuta el siguiente comando:

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Para ejecutar un trabajo de RAPIDS para Spark

1. Envíe un trabajo de RAPIDS para Spark a su clúster de Amazon EMR en EKS. En el siguiente código, se muestra un ejemplo de comando para iniciar el trabajo. La primera vez que ejecute el trabajo, es posible que tarde unos minutos en descargar la imagen y almacenarla en caché en el nodo.

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments": ["10000"],
"sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration": [{"classification":
"spark-defaults","properties": {"spark.executor.instances":
"2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
{"cloudWatchMonitoringConfiguration": {"logGroupName": "LOG_GROUP
_NAME"}, "s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

2. Para validar que el acelerador de RAPIDS para Spark esté activado, consulte los registros de los controladores de Spark. Estos registros se almacenan en CloudWatch o en la ubicación de S3 que especifique al ejecutar el start-job-run comando. El siguiente ejemplo muestra en general cómo se ven las líneas de registro:

```
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator build:
{version=22.08.0-amzn-0, user=release, url=, date=2022-11-03T03:32:45Z, revision=,
cudf_version=22.08.0, branch=}
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator JNI build:
{version=22.08.0, user=, url=https://github.com/NVIDIA/spark-rapids-jni.git,
date=2022-08-18T04:14:34Z, revision=a1b23cd_sample, branch=HEAD}
22/11/15 00:12:44 INFO RapidsPluginUtils: cudf build: {version=22.08.0,
user=, url=https://github.com/rapidsai/cudf.git, date=2022-08-18T04:14:34Z,
revision=a1b23ce_sample, branch=HEAD}
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator 22.08.0-amzn-0 using
cudf 22.08.0.
22/11/15 00:12:44 WARN RapidsPluginUtils:
spark.rapids.sql.multiThreadedRead.numThreads is set to 20.
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator is enabled, to disable
GPU support set `spark.rapids.sql.enabled` to false.
```

```
22/11/15 00:12:44 WARN RapidsPluginUtils: spark.rapids.sql.explain is set to
`NOT_ON_GPU`. Set it to 'NONE' to suppress the diagnostics logging about the query
placement on the GPU.
```

3. Para ver las operaciones que se ejecutarán en una GPU, siga estos pasos para habilitar el registro adicional. Tenga en cuenta la configuración “`spark.rapids.sql.explain : ALL`”.

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments": ["10000"],
"sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration":
[{"classification": "spark-defaults","properties":
{"spark.rapids.sql.explain":"ALL","spark.executor.instances":
"2","spark.executor.memory": "2G"}]], "monitoringConfiguration":
{"cloudWatchMonitoringConfiguration": {"logGroupName":
"LOG_GROUP_NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

El comando anterior es un ejemplo de un trabajo que utiliza la GPU. Su resultado se parecería al del ejemplo siguiente. Consulte esta clave como ayuda para entender el resultado:

- *: marca una operación que funciona en una GPU
- !: marca una operación que no se puede ejecutar en una GPU
- @: marca una operación que funciona en una GPU, pero que no se puede ejecutar porque está incluida en un plan que no se puede ejecutar en una GPU

```
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 118.64 ms
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 4.20 ms
22/11/15 01:22:58 INFO GpuOverrides: GPU plan transition optimization took 8.37 ms
22/11/15 01:22:59 WARN GpuOverrides:
  *Exec <ProjectExec> will run on GPU
    *Expression <Alias> substring(cast(date#149 as string), 0, 7) AS month#310
will run on GPU
    *Expression <Substring> substring(cast(date#149 as string), 0, 7) will run
on GPU
    *Expression <Cast> cast(date#149 as string) will run on GPU
```

```

*Exec <SortExec> will run on GPU
*Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
*Exec <ShuffleExchangeExec> will run on GPU
*Partitioning <RangePartitioning> will run on GPU
*Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
*Exec <UnionExec> will run on GPU
!Exec <ProjectExec> cannot run on GPU because not all expressions can
be replaced
    @Expression <AttributeReference> customerID#0 could run on GPU
    @Expression <Alias> Charge AS kind#126 could run on GPU
        @Expression <Literal> Charge could run on GPU
        @Expression <AttributeReference> value#129 could run on GPU
        @Expression <Alias> add_months(2022-11-15, cast(-(cast(_we0#142 as
bigint) + last_month#128L) as int)) AS date#149 could run on GPU
            ! <AddMonths> add_months(2022-11-15, cast(-
(cast(_we0#142 as bigint) + last_month#128L) as int)) cannot run
on GPU because GPU does not currently support the operator class
org.apache.spark.sql.catalyst.expressions.AddMonths
                @Expression <Literal> 2022-11-15 could run on GPU
                @Expression <Cast> cast(-(cast(_we0#142 as bigint) +
last_month#128L) as int) could run on GPU
                    @Expression <UnaryMinus> -(cast(_we0#142 as bigint) +
last_month#128L) could run on GPU
                        @Expression <Add> (cast(_we0#142 as bigint) +
last_month#128L) could run on GPU
                            @Expression <Cast> cast(_we0#142 as bigint) could run on
GPU
                                @Expression <AttributeReference> _we0#142 could run on
GPU
                                    @Expression <AttributeReference> last_month#128L could run
on GPU

```

Uso de la integración de Amazon Redshift para Apache Spark en Amazon EMR en EKS

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. De esta forma, puede usar Spark en Amazon EMR en EKS para procesar los datos almacenados en Amazon Redshift. La integración se basa en el [conector de código abierto spark-redshift](#). Para Amazon EMR 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](#)
- [Autenticación con la integración de Amazon Redshift para Apache Spark](#)
- [Lectura y escritura desde y hacia Amazon Redshift](#)
- [Consideraciones y limitaciones al utilizar el conector de Spark](#)

Lanzamiento de una aplicación de Spark mediante la integración de Amazon Redshift para Apache Spark

Para usar la integración, debe pasar las dependencias de Spark Redshift requeridas con su trabajo de Spark. Debe utilizar `--jars` para incluir bibliotecas relacionadas con el conector de Redshift. Para ver otras ubicaciones de archivos compatibles con la opción `--jars`, consulte la sección [Administración avanzada de dependencias](#) de la documentación de Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Para lanzar una aplicación Spark con la integración de Amazon Redshift para Apache Spark en Amazon EMR en EKS 6.9.0 o versiones posteriores, utilice el siguiente comando de ejemplo. Tenga en cuenta que las rutas enumeradas con la opción `--conf spark.jars` son las rutas predeterminadas para los archivos JAR.

```
aws emr-containers start-job-run \  
  
--virtual-cluster-id cluster_id \  
--execution-role-arn arn \  
--release-label emr-6.9.0-latest \  
--job-driver '{  
    "sparkSubmitJobDriver": {  
        "entryPoint": "s3://script_path",  
        "sparkSubmitParameters":  
            "--conf spark.kubernetes.file.upload.path=s3://upload_path  
            --conf spark.jars=
```

```

        /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
        /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
        /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
        /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar"
    }
}'

```

Autenticación con la integración de Amazon Redshift para Apache Spark

En las siguientes secciones se muestran las opciones de autenticación con Amazon Redshift al realizar la integración con Apache Spark. Las secciones muestran cómo recuperar las credenciales de inicio de sesión y también detalles sobre el uso del controlador JDBC con la autenticación de IAM.

AWS Secrets Manager Úselo para recuperar credenciales y conectarse a Amazon Redshift

Puede almacenar credenciales en Secrets Manager para autenticarse de forma segura en Amazon Redshift. Puede hacer que su trabajo de Spark llame a la API GetSecretValue para obtener las credenciales:

```

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

```


Utilizar la autenticación basada en IAM con el rol de ejecución de trabajos de Amazon EMR en EKS

A partir de la versión 6.9.0 de Amazon EMR en EKS, 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, puede especificar un esquema `jdbc:redshift:iam://`. Esto ordena al controlador de JDBC que utilice su rol de ejecución de trabajos de Amazon EMR en EKS para obtener las credenciales automáticamente.

Para obtener más información, consulte [Configurar una conexión de JDBC u ODBC para usar credenciales de IAM](#) en la Guía de administración de Amazon Redshift.

En el siguiente ejemplo de URL se utiliza un esquema `jdbc:redshift:iam://`.

```
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
<code>redshift:GetClusterCredentials</code>	Obligatorio para que el controlador de JDBC obtenga las credenciales de Amazon Redshift
<code>redshift:DescribeCluster</code>	Obligatorio si especifica el clúster de Amazon Redshift y la Región de AWS en la URL de JDBC en lugar del punto de conexión
<code>redshift-serverless:GetCredentials</code>	Obligatorio para que el controlador de JDBC obtenga las credenciales de Amazon Redshift sin servidor
<code>redshift-serverless:GetWorkgroup</code>	Obligatorio si utiliza Amazon Redshift sin servidor y especifica la URL en términos de nombre y región del grupo de trabajo

Su política de roles de ejecución de trabajos debe tener los siguientes permisos.

```
{
    "Effect": "Allow",
    "Action": [
        "redshift:GetClusterCredentials",
        "redshift:DescribeCluster",
        "redshift-serverless:GetCredentials",
        "redshift-serverless:GetWorkgroup"
    ],
    "Resource": [
        "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbname:CLUSTER_NAME/DATABASE_NAME",
        "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbuser:DATABASE_NAME/USER_NAME"
    ]
}
```

Autenticarse en Amazon Redshift con un controlador de JDBC

Establecer el nombre de usuario y la contraseña dentro de la URL de JDBC

Para autenticar un trabajo de Spark en un clúster de Amazon Redshift, puede 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, puede usar 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
```

Lectura y escritura desde y hacia Amazon Redshift

Los siguientes ejemplos de código se utilizan PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con una API de fuente de datos y con SparkSQL.

Data source API

Se utiliza PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift con una API de fuente 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::accountID:role/roleName"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName_copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .mode("error") \
    .save()
```

SparkSQL

Se utiliza PySpark para leer y escribir datos de muestra desde y hacia una base de datos de Amazon Redshift mediante SparkSQL.

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
```

```

    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

bucket = "s3://path/for/temp/data"
tableName = "tableName" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {tableName} (country string, data string)
      USING io.github.spark_redshift_community.spark.redshift
      OPTIONS (dbtable '{tableName}', tempdir '{bucket}', url '{url}', aws_iam_role
      '{aws_iam_role_arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(tableName, overwrite=False)
df = spark.sql(f"SELECT * FROM {tableName}")
df.show()

```

Consideraciones y limitaciones al utilizar el conector de Spark

El conector Spark admite diversas formas de administrar las credenciales, configurar la seguridad y conectarse con otros AWS servicios. Familiarícese con las recomendaciones de esta lista para configurar una conexión funcional y resiliente.

- Recomendamos activar SSL para la conexión JDBC desde Spark en Amazon EMR a Amazon Redshift.
- Le recomendamos que administre las credenciales del clúster de Amazon Redshift en AWS Secrets Manager como práctica recomendada. Consulte [Utilización AWS Secrets Manager para recuperar credenciales para conectarse a Amazon Redshift](#) para ver un ejemplo.
- Le recomendamos 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 recomendamos que bloquee el acceso público al clúster de Amazon Redshift.
 - Le recomendamos que active el [registro de auditoría de Amazon Redshift](#).
 - Recomendamos activar el [cifrado en reposo de Amazon Redshift](#).
- Tenga en cuenta las siguientes recomendaciones para Amazon S3:
 - Recomendamos [bloquear el acceso público a los buckets de Amazon S3](#).
 - Recomendamos utilizar el [cifrado del servidor de Amazon S3](#) para cifrar los buckets de Amazon S3 que utilice.
 - Recomendamos utilizar las [políticas de ciclo de vida de Amazon S3](#) para definir las reglas de retención del bucket de S3.
 - Amazon EMR siempre verifica el código importado desde el código abierto a la imagen. Por motivos de seguridad, no admitimos la codificación de claves de AWS acceso en el `tempdir` URI como método de autenticación de Spark a Amazon S3.

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

Uso de Volcano como programador personalizado para Apache Spark en Amazon EMR en EKS

Con Amazon EMR en EKS, puede usar el operador de Spark o `spark-submit` para ejecutar trabajos de Spark con los programadores personalizados de Kubernetes. Este tutorial explica cómo ejecutar trabajos de Spark con un programador de Volcano en una cola personalizada.

Descripción general

[Volcano](#) puede ayudar a administrar la programación de Spark con funciones avanzadas, como la programación de colas, la programación de reparto equitativo y la reserva de recursos. Para

obtener más información sobre las ventajas de Volcano, consulte [Por qué Spark elige Volcano como programador de lotes integrado en Kubernetes](#) en el blog de CNCF de The Linux Foundation.

Instalar y configurar Volcano

1. Elija uno de los siguientes comandos kubectl para instalar Volcano, en función de sus necesidades arquitectónicas:

```
# x86_64
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development.yaml
# arm64:
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development-arm64.yaml
```

2. Prepare un ejemplo de cola de Volcano. Una cola es una colección de [PodGroups](#). La cola adopta FIFO y es la base de la división de recursos.

```
cat << EOF > volcanoQ.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: Queue
metadata:
  name: sparkqueue
spec:
  weight: 4
  reclaimable: false
  capability:
    cpu: 10
    memory: 20Gi
EOF

kubectl apply -f volcanoQ.yaml
```

3. Cargue un PodGroup manifiesto de muestra en Amazon S3. PodGroup es un grupo de pods con una fuerte asociación. Por lo general, se utiliza un PodGroup para la programación por lotes. Envíe el siguiente ejemplo PodGroup a la cola que definió en el paso anterior.

```
cat << EOF > podGroup.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
spec:
  # Set minMember to 1 to make a driver pod
```

```

minMember: 1
# Specify minResources to support resource reservation.
# Consider the driver pod resource and executors pod resource.
# The available resources should meet the minimum requirements of the Spark job
# to avoid a situation where drivers are scheduled, but they can't schedule
# sufficient executors to progress.
minResources:
  cpu: "1"
  memory: "1Gi"
# Specify the queue. This defines the resource queue that the job should be
submitted to.
queue: sparkqueue
EOF

aws s3 mv podGroup.yaml s3://bucket-name

```

Ejecute una aplicación de Spark con el programador de Volcano con el operador de Spark

1. Si aún no lo ha hecho, complete los pasos de las secciones siguientes para configurarlo todo:
 - a. [Instalar y configurar Volcano](#)
 - b. [Configuración del operador de Spark para Amazon EMR en EKS](#)
 - c. [Instalar el operador de Spark](#)

Cuando ejecute el comando `helm install spark-operator-demo`, incluya los siguientes argumentos:

```

--set batchScheduler.enable=true
--set webhook.enable=true

```

2. Cree un archivo de definición `spark-pi.yaml` de `SparkApplication` con `batchScheduler` configurado.

```

apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:

```

```
type: Scala
mode: cluster
image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
imagePullPolicy: Always
mainClass: org.apache.spark.examples.SparkPi
mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
sparkVersion: "3.3.1"
batchScheduler: "volcano" #Note: You must specify the batch scheduler name as
'volcano'
restartPolicy:
  type: Never
volumes:
  - name: "test-volume"
    hostPath:
      path: "/tmp"
      type: Directory
driver:
  cores: 1
  coreLimit: "1200m"
  memory: "512m"
  labels:
    version: 3.3.1
  serviceAccount: emr-containers-sa-spark
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
executor:
  cores: 1
  instances: 1
  memory: "512m"
  labels:
    version: 3.3.1
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
```

- Envíe la aplicación de Spark con el siguiente comando. Esto también crea un objeto SparkApplication llamado spark-pi:

```
kubectl apply -f spark-pi.yaml
```

- Compruebe los eventos del objeto SparkApplication con el siguiente comando:


```
kubectl describe pods spark-pi-driver --namespace spark-operator
```

El primer evento de pod mostrará que Volcano ha programado los pods:

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	Scheduled	23s	volcano	Successfully assigned default/spark-pi-driver to integration-worker2

Ejecute una aplicación de Spark con el programador de Volcano con **spark-submit**

1. En primer lugar, complete los pasos de la sección [Configuración de spark-submit para Amazon EMR en EKS](#). Debe crear su distribución spark-submit con el soporte de Volcano. Para obtener más información, consulte la sección Crear de [Cómo usar Volcano como programador personalizado para Spark en Kubernetes](#) de la documentación de Apache Spark.
2. Establezca los valores de las siguientes variables de entorno:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Envíe la aplicación de Spark con el siguiente comando:

```
$SPARK_HOME/bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master $MASTER_URL \
  --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest \
  --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
  --deploy-mode cluster \
  --conf spark.kubernetes.namespace=spark-operator \
  --conf spark.kubernetes.scheduler.name=volcano \
  --conf spark.kubernetes.scheduler.volcano.podGroupTemplateFile=/path/to/podgroup-template.yaml \
  --conf
  spark.kubernetes.driver.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFeatu
  \
```

```
--conf
spark.kubernetes.executor.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFea
\
local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Compruebe los eventos del objeto SparkApplication con el siguiente comando:

```
kubectl describe pod spark-pi --namespace spark-operator
```

El primer evento de pod mostrará que Volcano ha programado los pods:

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	Scheduled	23s	volcano	Successfully assigned default/spark-
pi-driver	to integration-worker2			

YuniKorn Utilización como programador personalizado para Apache Spark en Amazon EMR en EKS

Con Amazon EMR en EKS, puede usar el operador de Spark o spark-submit para ejecutar trabajos de Spark con los programadores personalizados de Kubernetes. En este tutorial se explica cómo ejecutar tareas de Spark con un YuniKorn programador según una programación personalizada de colas y grupos.

Descripción general

[Apache YuniKorn](#) puede ayudarte a gestionar la programación de Spark con una programación compatible con las aplicaciones, de forma que puedas tener un control pormenorizado de las cuotas y prioridades de los recursos. Con la programación en grupo, solo se YuniKorn puede programar una aplicación cuando se puede satisfacer la solicitud mínima de recursos de la aplicación. Para obtener más información, consulte [Qué es la programación de pandillas](#) en el sitio de YuniKorn documentación de Apache.

Cree su clúster y prepárese para YuniKorn

Siga estos pasos para implementar un clúster de Amazon EKS. Puede cambiar la Región de AWS (region) y las zonas de disponibilidad (availabilityZones).

1. Defina el clúster de Amazon EKS:

```
cat <<EOF >eks-cluster.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: emr-eks-cluster
  region: eu-west-1

vpc:
  clusterEndpoints:
    publicAccess: true
    privateAccess: true

iam:
  withOIDC: true

nodeGroups:
- name: spark-jobs
  labels: { app: spark }
  instanceType: m5.xlarge
  desiredCapacity: 2
  minSize: 2
  maxSize: 3
  availabilityZones: ["eu-west-1a"]
EOF
```

2. Cree el clúster:

```
eksctl create cluster -f eks-cluster.yaml
```

3. Cree el espacio de nombres spark-job en el que ejecutará el trabajo de Spark:

```
kubectl create namespace spark-job
```

4. A continuación, cree un rol de Kubernetes y una vinculación de roles. Esto es obligatorio para la cuenta de servicio que utiliza la ejecución del trabajo de Spark.

a. Defina la cuenta de servicio, el rol y la vinculación de roles de los trabajos de Spark.

```
cat <<EOF >emr-job-execution-rbac.yaml
```

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: spark-sa
  namespace: spark-job
automountServiceAccountToken: false
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: spark-role
  namespace: spark-job
rules:
  - apiGroups: [ "", "batch", "extensions" ]
    resources: [ "configmaps", "serviceaccounts", "events", "pods", "pods/
exec", "pods/log", "pods/
portforward", "secrets", "services", "persistentvolumeclaims" ]
    verbs: [ "create", "delete", "get", "list", "patch", "update", "watch" ]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-sa-rb
  namespace: spark-job
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: spark-role
subjects:
  - kind: ServiceAccount
    name: spark-sa
    namespace: spark-job
EOF
```

- b. Aplique el rol de Kubernetes y la definición de la vinculación de roles con el siguiente comando:

```
kubectl apply -f emr-job-execution-rbac.yaml
```

Instalar y configurar YuniKorn

1. Use el siguiente comando kubectl para crear un espacio de nombres de yunikorn para implementar el programador de YuniKorn:

```
kubectl create namespace yunikorn
```

2. Para instalar el programador, ejecute los siguientes comandos de Helm:

```
helm repo add yunikorn https://apache.github.io/yunikorn-release
```

```
helm repo update
```

```
helm install yunikorn yunikorn/yunikorn --namespace yunikorn
```

Ejecute una aplicación de Spark con un YuniKorn programador con el operador de Spark

1. Si aún no lo ha hecho, complete los pasos de las secciones siguientes para configurarlo todo:
 - a. [Cree su clúster y prepárese para YuniKorn](#)
 - b. [Instalar y configurar YuniKorn](#)
 - c. [Configuración del operador de Spark para Amazon EMR en EKS](#)
 - d. [Instalar el operador de Spark](#)

Cuando ejecute el comando `helm install spark-operator-demo`, incluya los siguientes argumentos:

```
--set batchScheduler.enable=true  
--set webhook.enable=true
```

2. Cree un archivo de definición `spark-pi.yaml` de `SparkApplication`.

Para utilizarlo YuniKorn como planificador de sus trabajos, debe añadir determinadas anotaciones y etiquetas a la definición de la aplicación. Las anotaciones y etiquetas especifican la cola del trabajo y la estrategia de programación que desee utilizar.

En el siguiente ejemplo, la anotación `schedulingPolicyParameters` configura la planificación por grupos de la aplicación. A continuación, en el ejemplo se crean grupos de tareas para especificar la capacidad mínima que debe estar disponible antes de programar los pods para iniciar la ejecución del trabajo. Por último, especifica en la definición del grupo de tareas el uso de grupos de nodos con la etiqueta `"app": "spark"`, tal como se define en la sección [Cree su clúster y prepárese para YuniKorn](#).

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-job
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/schedulingPolicyParameters: "placeholderTimeoutSeconds=30
gangSchedulingStyle=Hard"
      yunikorn.apache.org/task-group-name: "spark-driver"
      yunikorn.apache.org/task-groups: |-
        [{
          "name": "spark-driver",
          "minMember": 1,
          "minResource": {
```

```

        "cpu": "1200m",
        "memory": "1Gi"
      },
      "nodeSelector": {
        "app": "spark"
      }
    },
    {
      "name": "spark-executor",
      "minMember": 1,
      "minResource": {
        "cpu": "1200m",
        "memory": "1Gi"
      },
      "nodeSelector": {
        "app": "spark"
      }
    }
  ]
  serviceAccount: spark-sa
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/task-group-name: "spark-executor"
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"

```

- Envíe la aplicación de Spark con el siguiente comando. Esto también crea un objeto SparkApplication llamado spark-pi:

```
kubectl apply -f spark-pi.yaml
```

- Compruebe los eventos del objeto SparkApplication con el siguiente comando:

```
kubectl describe sparkapplication spark-pi --namespace spark-job
```

El primer evento del pod mostrará que se YuniKorn han programado los pods:

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	Scheduling	3m12s	yunikorn	spark-operator/org-apache-spark-examples-sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal	GangScheduling	3m12s	yunikorn	Pod belongs to the taskGroup spark-driver, it will be scheduled as a gang member
Normal	Scheduled	3m10s	yunikorn	Successfully assigned spark
Normal	PodBindSuccessful	3m10s	yunikorn	Pod spark-operator/
Normal	TaskCompleted	2m3s	yunikorn	Task spark-operator/
Normal	Pulling	3m10s	kubelet	Pulling

Ejecuta una aplicación Spark con un YuniKorn programador con **spark-submit**

1. En primer lugar, complete los pasos de la sección [Configuración de spark-submit para Amazon EMR en EKS](#).
2. Establezca los valores de las siguientes variables de entorno:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Envíe la aplicación de Spark con el siguiente comando:

En el siguiente ejemplo, la anotación `schedulingPolicyParameters` configura la planificación por grupos de la aplicación. A continuación, en el ejemplo se crean grupos de tareas para especificar la capacidad mínima que debe estar disponible antes de programar los pods para iniciar la ejecución del trabajo. Por último, especifica en la definición del grupo de tareas el uso de grupos de nodos con la etiqueta `"app": "spark"`, tal como se define en la sección [Cree su clúster y prepárese para YuniKorn](#).

```
$SPARK_HOME/bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--master $MASTER_URL \
--conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-sa \
```



```
--deploy-mode cluster \
--conf spark.kubernetes.namespace=spark-job \
--conf spark.kubernetes.scheduler.name=yunikorn \
--conf spark.kubernetes.driver.annotation.yunikorn.apache.org/
schedulingPolicyParameters="placeholderTimeoutSeconds=30 gangSchedulingStyle=Hard"
\
--conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-group-
name="spark-driver" \
--conf spark.kubernetes.executor.annotation.yunikorn.apache.org/task-group-
name="spark-executor" \
--conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-groups='[{
    "name": "spark-driver",
    "minMember": 1,
    "minResource": {
        "cpu": "1200m",
        "memory": "1Gi"
    },
    "nodeSelector": {
        "app": "spark"
    }
},
{
    "name": "spark-executor",
    "minMember": 1,
    "minResource": {
        "cpu": "1200m",
        "memory": "1Gi"
    },
    "nodeSelector": {
        "app": "spark"
    }
}]' \
local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Compruebe los eventos del objeto SparkApplication con el siguiente comando:

```
kubectl describe pod spark-driver-pod --namespace spark-job
```

El primer evento de pods mostrará qué pods están programados: YuniKorn

Type	Reason	Age	From	Message
----	-----	----	----	-----

```
Normal Scheduling      3m12s yunikorn  spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling  3m12s yunikorn  Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled       3m10s yunikorn  Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn  Pod spark-operator/
Normal TaskCompleted   2m3s  yunikorn  Task spark-operator/
Normal Pulling         3m10s kubelet   Pulling
```

Seguridad de Amazon EMR en EKS

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted y usted. 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 AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los [AWS programas](#) de de . Para obtener más información sobre los programas de conformidad que se aplican a Amazon EMR, consulte [AWS Servicios dentro del alcance por programa de conformidad AWS Servicios incluidos en el ámbito por programa](#) .
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También eres responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon EMR en EKS. En los siguientes temas, se mostrará cómo configurar Amazon EMR en EKS para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger su Amazon EMR en los recursos de EKS.

Temas

- [Prácticas recomendadas de seguridad de Amazon EMR en EKS](#)
- [Protección de los datos](#)
- [Identity and Access Management](#)
- [Uso de Amazon EMR en EKS con AWS Lake Formation para un control de acceso detallado](#)
- [Registro y supervisión](#)
- [Uso de Amazon S3 Access Grants con Amazon EMR en EKS](#)
- [Validación de la conformidad de Amazon EMR en EKS](#)
- [Resiliencia de Amazon EMR en EKS](#)

- [Seguridad de la infraestructura de Amazon EMR en EKS](#)
- [Configuración y análisis de vulnerabilidades](#)
- [Conexión a Amazon EMR en EKS mediante un punto de conexión de VPC de interfaz](#)
- [Configuración del acceso entre cuentas de Amazon EMR en EKS](#)

Prácticas recomendadas de seguridad de Amazon EMR en EKS

Amazon EMR en EKS 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.

Note

Para conocer más prácticas recomendadas de seguridad, consulte [Prácticas recomendadas de seguridad de Amazon EMR en EKS](#).

Aplicar el principio de privilegio mínimo

Amazon EMR en EKS proporciona una política de acceso granular para las aplicaciones que utilizan roles de IAM, como los roles de ejecución. Estos roles de ejecución se asignan a las cuentas de servicio de Kubernetes mediante la política de confianza del rol de IAM. Amazon EMR en EKS crea pods dentro de un espacio de nombres de Amazon EKS registrado que ejecutan el código de aplicación proporcionado por el usuario. Los módulos de trabajo que ejecutan el código de la aplicación asumen la función de ejecución cuando se conectan a otros AWS servicios. Recomendamos que a los roles de ejecución solo se les otorguen los privilegios mínimos necesarios para el trabajo, como cubrir su aplicación y el acceso al destino del registro. También recomendamos auditar los trabajos para detectar permisos de forma regular y ante cualquier cambio en el código de la aplicación.

Lista de control de acceso para puntos de conexión

Los puntos de conexión administrados solo se pueden crear para los clústeres de EKS que se hayan configurado para utilizar al menos una subred privada en su VPC. Esta configuración restringe el

acceso a los equilibradores de carga creados por los puntos de conexión administrados para que solo se pueda acceder a ellos desde su VPC. Para mejorar aún más la seguridad, le recomendamos que configure los grupos de seguridad con estos equilibradores de carga para que puedan restringir el tráfico entrante a un conjunto seleccionado de direcciones IP.

Obtener las actualizaciones de seguridad más recientes para imágenes personalizadas

Para usar imágenes personalizadas con Amazon EMR en EKS, puede instalar cualquier binario y biblioteca en la imagen. Usted es responsable de los parches de seguridad de los archivos binarios que agregue a la imagen. Las imágenes de Amazon EMR en EKS se actualizan periódicamente con los últimos parches de seguridad. Para obtener la imagen más reciente, debe volver a crear las imágenes personalizadas siempre que haya una nueva versión de imagen base de la versión de Amazon EMR. Para obtener más información, consulte [Versiones de Amazon EMR en EKS](#) y [Información sobre cómo seleccionar un URI de imagen base](#).

Limitar el acceso a las credenciales del pod

Kubernetes admite varios métodos para asignar credenciales a un pod. El aprovisionamiento de varios proveedores de credenciales puede aumentar la complejidad del modelo de seguridad. Amazon EMR en EKS ha adoptado el uso de [roles de IAM para cuentas de servicios \(IRSA\)](#) como proveedor de credenciales estándar dentro de un espacio de nombres de EKS registrado. No se admiten otros métodos, como [kube2iam](#), [kiam](#) y el uso de un perfil de instancia de la EC2 instancia que se ejecuta en el clúster.

Aislar el código de aplicación no confiable

Amazon EMR en EKS no inspecciona la integridad del código de aplicación enviado por los usuarios del sistema. Si ejecuta un clúster virtual con varios inquilinos que está configurado con múltiples roles de ejecución y que puedan utilizar usuarios que no son de confianza y ejecutan código arbitrario para enviar trabajos, existe el riesgo de que una aplicación malintencionada aumente sus privilegios. En esta situación, considere la posibilidad de aislar los roles de ejecución con privilegios similares en un clúster virtual diferente.

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 Amazon EMR en los espacios de nombres administrados por EKS. Como mínimo,

no se deben conceder los siguientes permisos a los remitentes de trabajos en Amazon EMR en los espacios de nombres administrados por EKS.

- El RBAC de Kubernetes permite modificar el mapa de configuración, ya que Amazon EMR en EKS utiliza los mapas de configuración de Kubernetes para generar plantillas de pods administradas que tienen el nombre de la cuenta del servicio administrado. Este atributo no debe mutarse.
- Permisos de RBAC de Kubernetes para la ejecución en pods de Amazon EMR en EKS: para evitar dar acceso a plantillas de pods administradas que tienen el nombre de SA administrado. Este atributo no debe mutarse. Este permiso también puede dar acceso al token JWT montado en el pod, que luego se puede utilizar para recuperar las credenciales del rol de ejecución.
- Permisos RBAC de Kubernetes para crear pods, para evitar que los usuarios creen pods con un Kubernetes ServiceAccount que pueda estar asignado a una función de IAM con más privilegios que el usuario. AWS
- Permisos RBAC de Kubernetes para implementar un webhook mutante, a fin de evitar que los usuarios usen el webhook mutante para mutar el nombre de Kubernetes para los pods creados por Amazon EMR en EKS. ServiceAccount
- El RBAC de Kubernetes permite leer los secretos de Kubernetes, a fin de impedir que los usuarios lean los datos confidenciales almacenados en dichos secretos.

Restringir el acceso a las credenciales del perfil de instancia o rol de IAM del grupo de nodos

- Le recomendamos que asigne permisos mínimos a las funciones de IAM del grupo de nodos. AWS Esto ayuda a evitar el aumento de privilegios por parte de código que pueda ejecutarse con las credenciales del perfil de instancia de los nodos de trabajo de EKS.
- Para bloquear por completo el acceso a las credenciales del perfil de instancia a todos los pods que se ejecutan en los espacios de nombres administrados por Amazon EMR en EKS, le recomendamos que ejecute comandos iptables en los nodos de EKS. Para obtener más información, consulta [Restringir el acceso a las credenciales del perfil de EC2 instancia de Amazon](#). Sin embargo, es importante que establezca correctamente el ámbito de los roles de IAM de las cuentas de servicio para que los pods tengan todos los permisos necesarios. Por ejemplo, al rol de IAM del nodo de trabajo se le asignan permisos para extraer imágenes de contenedor de Amazon ECR. Si a un pod no se le asignan esos permisos, no podrá extraer imágenes de contenedor de Amazon ECR. También es necesario actualizar el complemento CNI de la VPC.

Para obtener más información, consulte [Tutorial: actualización del complemento CNI de la VPC para utilizar los roles de IAM para las cuentas de servicio](#).

Protección de los datos

El [modelo de responsabilidad AWS compartida](#) se aplica a la protección de datos en Amazon EMR en EKS. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se basa toda la AWS nube. 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 las [Preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulte [la entrada del blog sobre el modelo de responsabilidad AWS compartida y el RGPD](#) en el blog AWS de seguridad.

Con fines de protección de datos, le recomendamos que proteja las credenciales de las AWS cuentas 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.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Recomendamos TLS 1.2 o una versión posterior.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados de AWS los servicios.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.
- Utilice las opciones de cifrado de Amazon EMR en EKS para cifrar datos en reposo y en tránsito.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información acerca de los puntos de conexión de FIPS disponibles, consulte [Estándar de procesamiento de la información federal \(FIPS\) 140-2](#).

Le recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, números de cuenta de sus clientes, en los campos de formato libre, como el

campo Nombre. Esto incluye cuando trabaja con Amazon EMR en EKS u otros AWS servicios mediante la consola, la API o. AWS CLI AWS SDKs Es posible que cualquier dato que ingrese en Amazon EMR en EKS o en 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 las claves administradas por AWS Key Management Service, las claves administradas por Amazon S3 y las claves y certificados de los proveedores personalizados que usted suministre. Si AWS KMS lo utilizas como proveedor de claves, se aplican cargos por el almacenamiento y el uso de las claves de cifrado. Para obtener más información, consulte [AWS KMS Precios](#).

Antes de especificar las opciones de cifrado, decida qué sistemas de administración de claves y certificados quiere usar. A continuación, cree las claves y los certificados para los proveedores personalizados que especifique como parte de la configuración de cifrado.

Cifrado en reposo para datos de EMRFS en Amazon S3

El cifrado de Amazon S3 funciona con objetos del sistema de archivos de EMR (EMRFS) que se leen y se escriben en Amazon S3. Se especifica 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. También puede especificar métodos de cifrado diferentes para buckets individuales utilizando Per bucket encryption overrides (Reemplazos de cifrado por bucket). Independientemente de si el cifrado de Amazon S3 está 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. Para obtener más información detallada sobre cómo lleva a cabo Amazon S3 el cifrado, consulte [Protección de datos mediante cifrado](#) en la Guía para desarrolladores de Amazon Simple Storage Service.

Note

Al utilizarlas AWS KMS, se cobran cargos por el almacenamiento y el uso de las claves de cifrado. Para obtener más información, consulte [AWS KMS Precios](#).

Cifrado del servidor de Amazon S3

Cuando configura el cifrado del servidor de Amazon S3, Amazon S3 cifra los datos del objeto a medida que escribe los datos en el disco y descifra los datos cuando se accede. Para obtener más 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 en EKS:

- SSE-S3: Amazon S3 administra las claves en su nombre.
- SSE-KMS - Se utiliza an AWS KMS key para configurar políticas adecuadas para Amazon EMR en EKS.

SSE con claves proporcionadas por el cliente (SSE-C) no está disponible para su uso con Amazon EMR en EKS.

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 en su clúster. 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. 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.

Note

El CSE de Amazon S3 solo garantiza que los datos de EMRFS intercambiados con Amazon S3 se cifren; no se cifran todos los datos en volúmenes de instancias de clúster. Además, ya que Hue no utiliza EMRFS, los objetos que Hue S3 File Browser escribe en Amazon S3 no se cifran.

Cifrado de disco local

Apache Spark admite el cifrado de datos temporales escritos en discos locales. Esto cubre archivos aleatorios, derrames aleatorios y bloques de datos almacenados en el disco para variables de transmisión y almacenamiento en caché. No cubre el cifrado de los datos de salida generados por aplicaciones con APIs, por ejemplo, `saveAsHadoopFile` o `saveAsTable`. Es posible que tampoco abarque los archivos temporales creados explícitamente por el usuario. Para obtener más información, consulte [Cifrado de almacenamiento local](#) en la documentación de Spark. Spark no admite datos cifrados en un disco local, como los datos intermedios que un proceso ejecutor escribe en un disco local cuando los datos no caben en la memoria. Los datos que se conservan en el disco se asignan al tiempo de ejecución del trabajo, y Spark genera dinámicamente la clave que se usa para cifrar los datos de cada ejecución del trabajo. Una vez que termina el trabajo de Spark, ningún otro proceso puede descifrar los datos.

En el caso de los pods controladores y ejecutores, se cifran los datos en reposo que se conservan en el volumen montado. [Hay tres opciones diferentes de almacenamiento AWS nativo que puedes usar con Kubernetes: EBS, FSx EFS y para Lustre](#). Las tres ofrecen cifrado en reposo mediante una clave administrada por el servicio o una AWS KMS key. Para obtener más información, consulte la [Guía de prácticas recomendadas de EKS](#). Con este enfoque, se cifran todos los datos conservados en el volumen montado.

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 información adicional, consulte Rotación. AWS KMS keys](#)

Cifrado en tránsito

Hay habilitados diversos mecanismos de cifrado con el cifrado en tránsito. Se trata de características de código abierto, específicas de la aplicación y que pueden variar según la versión de Amazon EMR en EKS. Las siguientes características de cifrado específicas de la aplicación se pueden habilitar con Amazon EMR en EKS:

- Spark
 - Las comunicaciones RPC internas entre componentes Spark, como el servicio de transferencia de bloques y el servicio de reorganización externo, se cifran mediante el cifrado AES-256 en las versiones 5.9.0 y posteriores de Amazon EMR. En versiones anteriores, la comunicación RPC interna se cifraba mediante SASL con DIGEST-MD5 como sistema de cifrado.

- Las comunicaciones del protocolo HTTP con interfaces de usuario como Spark History Server y servidores de archivos compatibles con HTTPS se cifran mediante la configuración SSL de Spark. Para obtener más información, consulte [SSL Configuration](#) en la documentación de Spark.

Para obtener más información, consulte [Configuración de seguridad de Spark](#).

- Debe permitir únicamente las conexiones cifradas a través de HTTPS (TLS) mediante [la SecureTransport condición aws:](#) de las políticas de IAM de bucket de Amazon S3.
- Los resultados de las consultas que se envían a clientes JDBC u ODBC se cifran mediante TLS.

Identity and Access Management

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de Amazon EMR en EKS. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- [Público](#)
- [Autenticación con identidades](#)
- [Administración de acceso mediante políticas](#)
- [Cómo funciona Amazon EMR en EKS con IAM](#)
- [Uso de roles vinculados a servicios para Amazon EMR en EKS](#)
- [Políticas administradas para Amazon EMR en EKS](#)
- [Uso de roles de ejecución de trabajos con Amazon EMR en EKS](#)
- [Ejemplos de políticas basadas en identidades para Amazon EMR en EKS](#)
- [Políticas para el control de acceso basado en etiquetas](#)
- [Solución de problemas de identidad y acceso de Amazon EMR en EKS](#)

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que realice en Amazon EMR en EKS.

Usuario de servicio: si utiliza el servicio Amazon EMR en EKS para llevar a cabo su trabajo, su administrador proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Amazon EMR en EKS para llevar a cabo su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos al administrador. Si no puede acceder a una característica de Amazon EMR en EKS, consulte [Solución de problemas de identidad y acceso de Amazon EMR en EKS](#).

Administrador de servicio: si está a cargo de los recursos de Amazon EMR en EKS de su empresa, probablemente tenga acceso completo a Amazon EMR en EKS. Su trabajo consiste en determinar a qué características y recursos de Amazon EMR en EKS deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su gestor de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo la empresa puede utilizar IAM con Amazon EMR en EKS, consulte [Cómo funciona Amazon EMR en EKS con IAM](#).

Administrador de IAM: si es administrador de IAM, es posible que desee obtener información sobre cómo escribir políticas para administrar el acceso a Amazon EMR en EKS. Para consultar ejemplos de políticas de Amazon EMR en EKS basadas en identidades que puede utilizar en IAM, consulte [Ejemplos de políticas basadas en identidades para Amazon EMR en EKS](#).

Autenticación con identidades

La autenticación es la forma de iniciar sesión para AWS usar sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su gestor habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte [Cómo iniciar sesión Cuenta de AWS en su](#) Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus

credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre la firma de solicitudes, consulte [AWS Signature Versión 4 para solicitudes API](#) en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte [Autenticación multifactor](#) en la Guía del usuario de AWS IAM Identity Center y [Autenticación multifactor AWS en IAM](#) en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utiliza el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulta [Tareas que requieren credenciales de usuario raíz](#) en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utiliza AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulta [¿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 propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulta [Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración](#) en la Guía del usuario de IAM.

Un [grupo de IAM](#) es una identidad que especifica un conjunto de usuarios de IAM. No puedes iniciar sesión como grupo. Puedes usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte [Casos de uso para usuarios de IAM](#) en la Guía del usuario de IAM.

Roles de IAM

Un [rol de IAM](#) es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una persona determinada. Para asumir temporalmente un rol de IAM en el AWS Management Console, puede [cambiar de un rol de usuario a uno de IAM](#) (consola). Puedes asumir un rol llamando a una operación de AWS API AWS CLI o usando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulta [Métodos para asumir un rol](#) en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puedes crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles de federación, consulte [Crear un rol para un proveedor de identidad de terceros \(federación\)](#) en la Guía de usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué

puedes acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulta [Conjuntos de permisos](#) en la Guía del usuario de AWS IAM Identity Center .

- **Permisos de usuario de IAM temporales:** un usuario de IAM puedes asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- **Acceso entre cuentas:** puedes utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, en algunos casos Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulta [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.
- **Acceso entre servicios:** algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
- **Sesiones de acceso directo (FAS):** cuando utilizas un usuario o un rol de IAM para realizar acciones en AWS ellas, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulta [Reenviar sesiones de acceso](#).
- **Rol de servicio:** un rol de servicio es un [rol de IAM](#) que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte [Creación de un rol para delegar permisos a un Servicio de AWS](#) en la Guía del usuario de IAM.
- **Función vinculada al servicio:** una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puedes asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puedes ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y realizan AWS CLI solicitudes a la AWS API. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte [Usar un rol de IAM para conceder permisos a las aplicaciones que se ejecutan en EC2 instancias de Amazon](#) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulta [Información general de políticas JSON](#) en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puedes crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puedes añadir las políticas de IAM a roles y los usuarios puedes asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utiliza para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puedes 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.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para obtener más información sobre cómo elegir una política administrada o una política insertada, 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 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 pueden realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#) en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No pueden usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACLs)

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios compatibles. AWS WAF ACLs Para obtener más información ACLs, consulte la [descripción general de la lista de control de acceso \(ACL\)](#) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- **Límites de permisos:** un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puedes conceder a una entidad de IAM (usuario o rol de IAM). Puedes establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulta [Límites de permisos para las entidades de IAM](#) en la Guía del usuario de IAM.
- **Políticas de control de servicios (SCPs):** SCPs son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations es un servicio para agrupar y administrar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una Usuario raíz de la cuenta de AWS. Para obtener más información sobre Organizations SCPs, consulte las [políticas de control de servicios](#) en la Guía del AWS Organizations usuario.
- **Políticas de control de recursos (RCPs):** RCPs son políticas de JSON que puedes usar para establecer los permisos máximos disponibles para los recursos de tus cuentas sin actualizar las políticas de IAM asociadas a cada recurso que poseas. El RCP limita los permisos de los recursos en las cuentas de los miembros y puede afectar a los permisos efectivos de las identidades, incluidos los permisos Usuario raíz de la cuenta de AWS, independientemente de si pertenecen a su organización. Para obtener más información sobre Organizations e RCPs incluir una lista de Servicios de AWS ese apoyo RCPs, consulte [Políticas de control de recursos \(RCPs\)](#) en la Guía del AWS Organizations usuario.
- **Políticas de sesión:** las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también puedes proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulta [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 saber cómo se AWS determina si se debe permitir una solicitud

cuando se trata de varios tipos de políticas, consulte la [lógica de evaluación de políticas](#) en la Guía del usuario de IAM.

Cómo funciona Amazon EMR en EKS con IAM

Antes de utilizar IAM para administrar el acceso a Amazon EMR en EKS, obtenga información sobre qué características de IAM se encuentran disponibles con Amazon EMR en EKS.

Características de IAM que puede utilizar con Amazon EMR en EKS

Característica de IAM	Soporte de Amazon EMR en EKS
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	Sí
ACLs	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 visión general de cómo Amazon EMR en EKS y otros AWS servicios funcionan con la mayoría de las funciones de IAM, consulte los [AWS servicios que funcionan con IAM en la Guía del usuario de IAM](#).

Políticas basadas en identidades para Amazon EMR en EKS

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. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está asociada. 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.

Ejemplos de políticas basadas en identidades para Amazon EMR en EKS

Para ver ejemplos de políticas basadas en identidades para Amazon EMR en EKS, consulte [Ejemplos de políticas basadas en identidades para Amazon EMR en EKS](#).

Políticas basadas en recursos de Amazon EMR en EKS

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 puedes utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puedes realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#) en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política basada en recursos concede acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional.

Para obtener más información, consulte [Cross account resource access in IAM](#) en la Guía del usuario de IAM.

Acciones de políticas de Amazon EMR en EKS

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puedes utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista completa de acciones de políticas de Amazon EMR en EKS, consulte [Acciones, recursos y claves de condición de Amazon EMR en EKS](#) en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Amazon EMR en EKS utilizan el siguiente prefijo antes de la acción:

```
emr-containers
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [  
    "emr-containers:action1",  
    "emr-containers:action2"  
]
```

Para ver ejemplos de políticas basadas en identidades para Amazon EMR en EKS, consulte [Ejemplos de políticas basadas en identidades para Amazon EMR en EKS](#).

Recursos de políticas para Amazon EMR en EKS

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes 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. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el [Nombre de recurso de Amazon \(ARN\)](#). Puedes hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utiliza un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"

```

Para ver una lista de Amazon EMR sobre los tipos de recursos de EKS y sus tipos de recursos ARNs, consulte [Recursos definidos por Amazon EMR en EKS en](#) la Referencia de autorización de servicio. Para saber qué acciones puede especificar el ARN de cada recurso, consulte [Acciones, recursos y claves de condición de Amazon EMR en EKS](#).

Para ver ejemplos de políticas basadas en identidades para Amazon EMR en EKS, consulte [Ejemplos de políticas basadas en identidades para Amazon EMR en EKS](#).

Claves de condición de políticas para Amazon EMR en EKS

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. 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.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación AND lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una OR operación

lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puedes utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puedes conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulta [Elementos de la política de IAM: variables y etiquetas](#) en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de [contexto de condición AWS globales en la Guía](#) del usuario de IAM.

Para ver una lista de las claves de condición de Amazon EMR en EKS y saber qué acciones y recursos puede utilizar una clave de condición, consulte [Acciones, recursos y claves de condición de Amazon EMR en EKS](#) en la Referencia de autorizaciones de servicio.

Para ver ejemplos de políticas basadas en identidades para Amazon EMR en EKS, consulte [Ejemplos de políticas basadas en identidades para Amazon EMR en EKS](#).

Listas de control de acceso (ACLs) en Amazon EMR en EKS

Soporta ACLs: No

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs 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 Amazon EMR en EKS

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. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el [elemento de condición](#) de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información 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, consulta [Uso del control de acceso basado en atributos \(ABAC\)](#) en la Guía del usuario de IAM.

Uso de credenciales temporales con Amazon EMR en EKS

Compatibilidad con credenciales temporales: sí

Algunos Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluidas las que Servicios de AWS funcionan con credenciales temporales, consulta [Cómo Servicios de AWS funcionan con IAM](#) en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para obtener más información sobre el cambio de roles, consulte [Cambio de un usuario a un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte [Credenciales de seguridad temporales en IAM](#).

Permisos de entidades principales entre servicios para Amazon EMR en EKS

Admite sesiones de acceso directo (FAS): sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio

de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulta [Reenviar sesiones de acceso](#).

Roles de servicio para Amazon EMR en EKS

Compatible con roles de servicio	No

Roles vinculados a servicios para Amazon EMR en EKS

Compatible con roles vinculados al servicio	Sí

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulta [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 ver la documentación acerca del rol vinculado a servicios para ese servicio.

Uso de roles vinculados a servicios para Amazon EMR en EKS

[Amazon EMR en EKS utiliza funciones vinculadas a AWS Identity and Access Management servicios \(IAM\)](#). Un rol vinculado a servicios es un tipo único de rol de IAM que se encuentra vinculado directamente a Amazon EMR en EKS. Amazon EMR predefine las funciones vinculadas al servicio en EKS e incluyen todos los permisos que el servicio necesita para llamar a otros AWS servicios en su nombre.

Un rol vinculado a servicios simplifica la configuración de Amazon EMR en EKS porque ya no tendrá que agregar de forma manual los permisos necesarios. Amazon EMR en EKS define los permisos de sus roles vinculados a servicios y, a menos que esté definido de otra manera, solo Amazon EMR en EKS 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 Amazon EMR en EKS, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información acerca de otros servicios que son compatibles con roles vinculados a servicios, consulte [Servicios de AWS que funcionan con IAM](#) y busque los servicios que muestran Yes (Sí) en la columna Service-Linked Role (Rol vinculado a servicios). Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado a servicios en cuestión.

Permisos de roles vinculados a servicios para Amazon EMR en EKS

Amazon EMR en EKS usa el rol vinculado al servicio denominado.

`AWSServiceRoleForAmazonEMRContainers`

El rol vinculado al servicio `AWSServiceRoleForAmazonEMRContainers` depende de los siguientes servicios para asumir el rol:

- `emr-containers.amazonaws.com`

La política de permisos del rol `AmazonEMRContainersServiceRolePolicy` permite que Amazon EMR en EKS lleve a cabo las siguientes acciones en los recursos especificados, tal como demuestra la siguiente instrucción de política.

Note

El contenido de la política gestionada cambia, por lo que la política que se muestra aquí puede cambiar. out-of-date Consulta la mayoría up-to-date de la documentación sobre políticas en [Amazon EMRContainers ServiceRolePolicy](#) en la Guía de referencia de políticas AWS gestionadas.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:ListNodeGroups",
        "eks:DescribeNodeGroup",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
```

```

        "elasticloadbalancing:DescribeInstanceHealth",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "eks:ListPodIdentityAssociations",
        "eks:DescribePodIdentityAssociation"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "acm:ImportCertificate",
      "acm:AddTagsToCertificate"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/emr-container:endpoint:managed-certificate": "true"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "acm:DeleteCertificate"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/emr-container:endpoint:managed-certificate":
"true"
      }
    }
  }
]
}

```

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 servicios para Amazon EMR en EKS

No necesita crear manualmente un rol vinculado a servicios. Al crear un clúster virtual, Amazon EMR en EKS se encarga de crear el rol vinculado a servicios en su nombre.

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 un clúster virtual, Amazon EMR en EKS se encarga de crear de nuevo el rol vinculado a servicios en su nombre.

También puede utilizar la consola de IAM para crear un rol vinculado al servicio con el caso de uso de Amazon EMR en EKS. En la API AWS CLI o en la AWS API, crea un rol vinculado al servicio con el nombre del `emr-containers.amazonaws.com` servicio. 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 servicios para Amazon EMR en EKS

Amazon EMR en EKS no permite editar el rol vinculado a servicios de `AWSServiceRoleForAmazonEMRContainers`. Después de crear un rol vinculado al servicio, no podrá cambiar el nombre del rol, ya que varias entidades podrían hacer referencia al rol. Sin embargo, sí puede editar la descripción del rol con IAM. Para obtener más información, consulte [Edición de un rol vinculado a servicios](#) en la Guía del usuario de IAM.

Eliminar un rol vinculado a servicios para Amazon EMR en EKS

Si ya no necesita usar una característica o servicio que requieran un rol vinculado a un servicio, le recomendamos que elimine dicho rol. Así no tendrá una entidad no utilizada que no se supervise ni mantenga de forma activa. Sin embargo, debe limpiar los recursos de su rol vinculado al servicio antes de eliminarlo manualmente.

Note

Si el servicio de Amazon EMR en EKS utiliza el rol cuando intenta eliminar los recursos, la eliminación podría producir un error. En tal caso, espere unos minutos e intente de nuevo la operación.

Para eliminar los recursos de Amazon EMR en EKS que utiliza el

AWSServiceRoleForAmazonEMRContainers

1. Abra la consola de Amazon EMR.
2. Elija un clúster virtual.
3. En la página `Virtual Cluster`, elija `Eliminar`.
4. Repita este procedimiento para el resto de los clústeres virtuales de la cuenta.

Para eliminar manualmente el rol vinculado a servicios mediante IAM

Utilice la consola de IAM AWS CLI, la o la AWS API para eliminar la función vinculada al `AWSServiceRoleForAmazonEMRContainers` servicio. Para obtener más información, consulte [Eliminación de un rol vinculado a servicios](#) en la Guía del usuario de IAM.

Regiones admitidas para los roles vinculados a servicios de Amazon EMR en EKS

Amazon EMR en EKS admite el uso de roles vinculados a servicios en todas las regiones en las que se encuentra disponible el servicio. Para obtener más información, consulte [Puntos de conexión de servicio y cuotas de Amazon EMR en EKS](#).

Políticas administradas para Amazon EMR en EKS

Consulte los detalles sobre las actualizaciones de las políticas AWS gestionadas para Amazon EMR en EKS desde el 1 de marzo de 2021.

Cambio	Descripción	Fecha
AmazonEMRContainerServiceRolePolicy - Se han añadido permisos de lectura para enumerar las asociaciones de identidad de los pods de EKS en un clúster y otro permiso de lectura para mostrar información descriptiva sobre las	Se añaden los siguientes permisos a la política: <code>eks:ListPodIdentityAssociations</code> , <code>eks:DescribePodIdentityAssociation</code> .	3 de febrero de 2023

Cambio	Descripción	Fecha
asociaciones de identidad de los pods de un clúster. Para obtener más información, consulta Amazon EMRContainers ServiceRolePolicy .		
AmazonEMRContainerServiceRolePolicy : se agregaron permisos para describir y enumerar los grupos de nodos de Amazon EKS, describir los grupos objetivo del equilibrador de carga y describir el estado del destino del equilibrador de carga.	Se agregaron los siguientes permisos a la política: eks:ListNodeGroups , eks:DescribeNodeGroup , elasticloadbalancing:DescribeTargetGroups , elasticloadbalancing:DescribeTargetHealth .	13 de marzo de 2023
AmazonEMRContainerServiceRolePolicy - Se han añadido permisos para importar y eliminar certificados en AWS Certificate Manager.	Se agregaron los siguientes permisos a la política: acm:ImportCertificate , acm:AddTagsToCertificate , acm>DeleteCertificate .	3 de diciembre de 2021
Amazon EMR en EKS comenzó a hacer el seguimiento de los cambios.	Amazon EMR en EKS comenzó a realizar un seguimiento de los cambios en sus políticas AWS gestionadas.	1 de marzo de 2021

Uso de roles de ejecución de trabajos con Amazon EMR en EKS

Para usar el comando `StartJobRun` para enviar una ejecución de trabajo en un clúster de EKS, antes debe incorporar un rol de ejecución de trabajos para usarlo con un clúster virtual. Para obtener

más información, consulta [Crear un rol de ejecución de trabajos](#) en [Configuración de Amazon EMR en EKS](#). También puede seguir las instrucciones de la sección [Crear rol de IAM para la ejecución de trabajos](#) del taller de Amazon EMR en EKS.

Los siguientes permisos deben incluirse en la política de confianza del rol de ejecución de trabajos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::AWS_ACCOUNT_ID:oidc-provider/OIDC_PROVIDER"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-containers-sa-*-*AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME"
        }
      }
    }
  ]
}
```

La política de confianza del ejemplo anterior concede permisos únicamente a una cuenta de servicio de Kubernetes administrada por Amazon EMR con un nombre que coincida con el patrón `emr-containers-sa-*-*AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME`. Las cuentas de servicio que sigan este patrón se crearán automáticamente al enviar el trabajo y se circunscribirán al espacio de nombres en el que envíes el trabajo. Esta política de confianza permite a estas cuentas de servicio asumir el rol de ejecución y obtener las credenciales temporales del rol de ejecución. Las cuentas de servicio de un clúster de Amazon EKS diferente o de un espacio de nombres diferente dentro del mismo clúster de EKS no pueden asumir el rol de ejecución.

Puede ejecutar el siguiente comando para actualizar automáticamente la política de confianza en el formato indicado anteriormente.

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

```
--namespace namespace \
--role-name iam_role_name_for_job_execution
```

Control del acceso al rol de ejecución

Un administrador de su clúster de Amazon EKS puede crear un clúster virtual de Amazon EMR en EKS con varios inquilinos al que un administrador de IAM puede agregar varios roles de ejecución. Debido a que los inquilinos que no son de confianza pueden usar estos roles de ejecución para enviar trabajos que ejecutan código arbitrario, es posible que desee restringirlos para que no puedan ejecutar código que obtenga los permisos asignados a uno o más de estos roles de ejecución. Para restringir la política de IAM asociada a una identidad de IAM, el administrador de IAM puede utilizar la clave de condición opcional del nombre de recurso de Amazon (ARN) `emr-containers:ExecutionRoleArn`. Esta condición acepta una lista de roles de ejecución ARNs que tienen permisos para el clúster virtual, como se demuestra en la siguiente política de permisos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": "arn:aws:emr-containers:REGION:AWS_ACCOUNT_ID:/
virtualclusters/VIRTUAL_CLUSTER_ID",
      "Condition": {
        "ArnEquals": {
          "emr-containers:ExecutionRoleArn": [
            "execution_role_arn_1",
            "execution_role_arn_2",
            ...
          ]
        }
      }
    }
  ]
}
```

Si quiere permitir todos los roles de ejecución que comiencen con un prefijo concreto, como `MyRole`, puede sustituir el operador de condición `ArnEquals` por el operador `ArnLike` y sustituir el valor `execution_role_arn` de la condición por un carácter comodín `*`. Por ejemplo,

`arn:aws:iam::AWS_ACCOUNT_ID:role/MyRole*`. También se admiten todas las demás [claves de condición de ARN](#).

Note

Con Amazon EMR en EKS, no puede conceder permisos a roles de ejecución en función de etiquetas o atributos. Amazon EMR en EKS no admite el control de acceso basado en etiquetas (TBAC) ni el control de acceso basado en atributos (ABAC) para roles de ejecución.

Ejemplos de políticas basadas en identidades para Amazon EMR en EKS

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Amazon EMR en EKS. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o la AWS API. Un administrador de IAM puedes crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puedes añadir las políticas de IAM a roles y los usuarios puedes asumirlos.

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 en EKS, incluido el formato de cada uno de los tipos de recursos, consulte [Acciones, recursos y claves de condición de Amazon EMR en EKS en](#) la Referencia de autorización de servicios. ARNs

Temas

- [Prácticas recomendadas sobre las políticas](#)
- [Uso de la consola de Amazon EMR en EKS](#)
- [Cómo permitir a los usuarios consultar sus propios permisos](#)

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon EMR en EKS de su 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:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulta 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, consulta [Políticas y permisos en IAM](#) en la Guía del usuario de IAM.
- Utiliza condiciones en las políticas de IAM para restringir aún más el acceso: puedes 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 el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulta [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.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para 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.

Uso de la consola de Amazon EMR en EKS

Para acceder a la consola de Amazon EMR en EKS, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle mostrar y consultar los detalles sobre los recursos de Amazon EMR en EKS en la Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realizan llamadas a la API o a la AWS CLI API. AWS En su lugar, permite el acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles puedan seguir utilizando Amazon EMR en la consola de EKS, adjunte también la política `ReadOnly AWS gestionada ConsoleAccess` o Amazon EMR en EKS a las entidades. Para obtener más información, consulte [Adición de permisos a un usuario](#) en la Guía del usuario de IAM:

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas gestionadas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API o. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "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": "*"
}
]
```

Políticas para el control de acceso basado en etiquetas

Puede utilizar condiciones en su política basada en identidades para controlar el acceso a clústeres virtuales y ejecuciones de trabajos de EMR basadas en etiquetas. Para obtener más información acerca del etiquetado, consulte [Etiquetado de sus recursos de Amazon EMR en EKS](#).

En los siguientes ejemplos, se muestran distintos supuestos y formas de utilizar los operadores de condiciones con las claves de condición de Amazon EMR en EKS. 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 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 clústeres de Amazon EMR en EKS.

Permitir acciones solo en recursos con valores de etiqueta específicos

En el siguiente ejemplo de política, el operador de StringEquals condición intenta hacer coincidir dev con el valor del departamento de etiquetas. Si la etiqueta department no se agregó al clúster 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 clústeres virtuales que tengan esta etiqueta con este valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

También puede especificar varios valores de etiqueta utilizando un operador de condición. Por ejemplo, a fin de permitir que las acciones en clústeres virtuales donde la etiqueta department contiene el valor dev o test, podría sustituir el bloque de condición del ejemplo anterior por los siguientes.

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/department": ["dev", "test"]
  }
}
```

Requerir etiquetado cuando se crea un recurso

En el siguiente ejemplo, la etiqueta debe aplicarse al crear el clúster virtual.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

La siguiente instrucción de política permite a un usuario crear un clúster virtual solo si el clúster tiene una etiqueta department, que puede contener cualquier valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

Denegar el acceso para agregar y eliminar etiquetas

El efecto de esta política consiste en denegar a un usuario el permiso para agregar o eliminar cualquier etiqueta en clústeres virtuales que están etiquetados con una etiqueta `department` que contiene el valor `dev`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-containers:TagResource",
        "emr-containers:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Solución de problemas de identidad y acceso de Amazon EMR en EKS

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que es posible que surjan cuando se trabaja con Amazon EMR en EKS e IAM.

Temas

- [No tengo autorización para llevar a cabo una acción en Amazon EMR en EKS](#)
- [No estoy autorizado a realizar lo siguiente: PassRole](#)
- [Quiero permitir que personas ajenas a mi AWS cuenta accedan a mis recursos de Amazon EMR en EKS](#)

No tengo autorización para llevar a cabo una acción en Amazon EMR en EKS

Si AWS Management Console le indica que no está autorizado a realizar una acción, debe ponerse en contacto con el administrador para obtener ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

En el siguiente ejemplo, el error se produce cuando el usuario mateojackson intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *my-example-widget*, pero no tiene los permisos ficticios `emr-containers:GetWidget`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-containers: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-containers:GetWidget`.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para llevar a cabo la acción `iam:PassRole`, sus políticas deben actualizarse para permitirle pasar un rol a Amazon EMR en EKS.

Algunos Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un 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 llevar a cabo una acción en Amazon EMR en EKS. 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 administrador. AWS El gestor es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mi AWS cuenta accedan a mis recursos de Amazon EMR en EKS

Puedes crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puedes especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admiten políticas basadas en recursos o listas de control de acceso (ACLs), puede utilizar esas políticas para permitir que las personas accedan a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon EMR en EKS admite estas características, consulte [Cómo funciona Amazon EMR en EKS con IAM](#).
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte [Proporcionar acceso a un usuario de IAM en otro de su propiedad en la Cuenta de AWS Guía del usuario](#) de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta [Cómo proporcionar acceso a recursos que Cuentas de AWS 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.

Uso de Amazon EMR en EKS con AWS Lake Formation para un control de acceso detallado

Con la versión 7.7 y versiones posteriores de Amazon EMR, puede aprovechar AWS Lake Formation para aplicar controles de acceso detallados en las tablas del catálogo de datos de AWS Glue respaldadas por buckets de Amazon S3. Esta capacidad le permite configurar controles de acceso a nivel de tabla, fila, columna y celda para las consultas de lectura en Amazon EMR en EKS Spark Jobs.

Temas

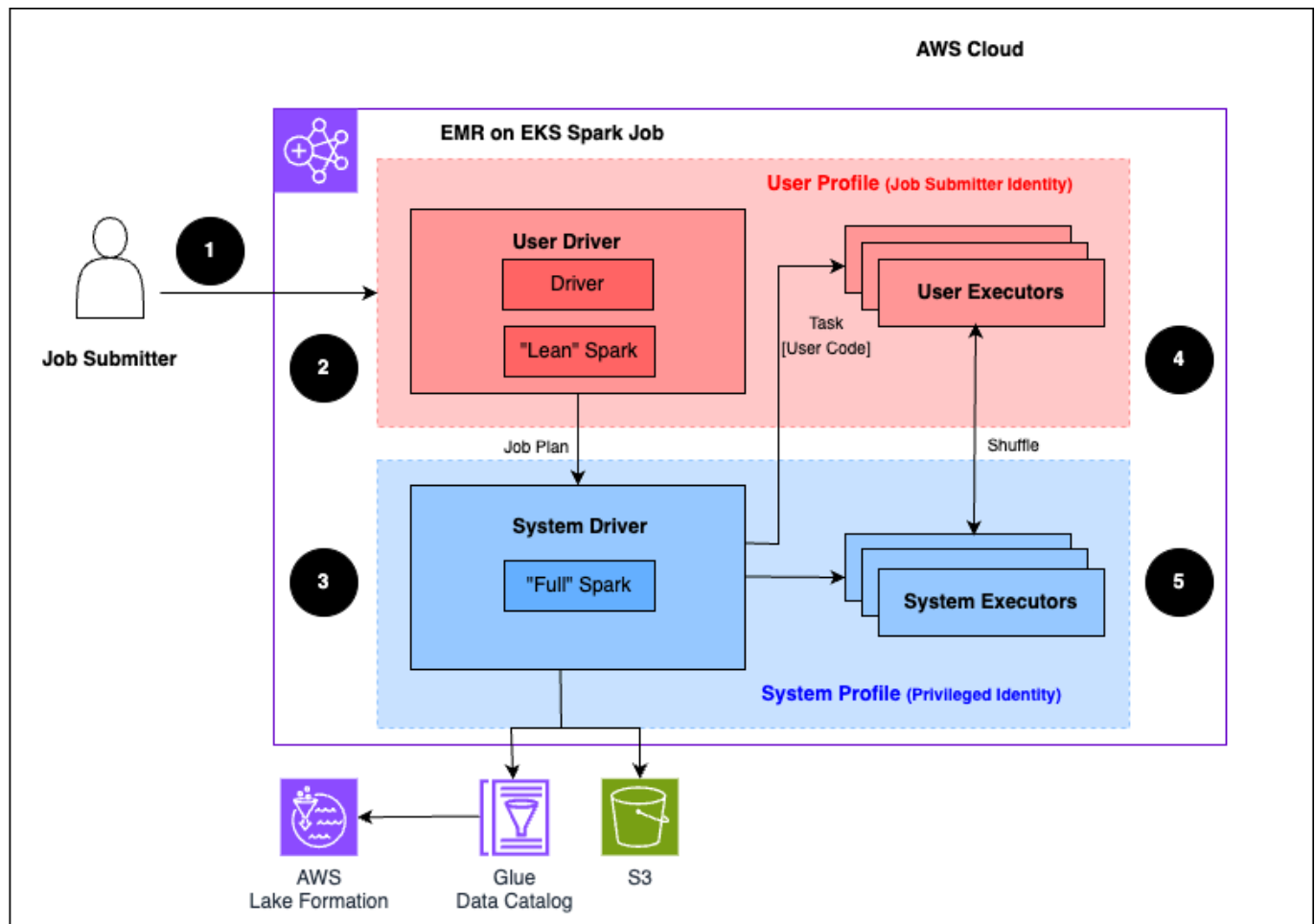
- [Cómo funciona Amazon EMR en EKS con Lake Formation AWS](#)

- [Habilite Lake Formation con Amazon EMR en EKS](#)
- [Consideraciones y limitaciones](#)
- [Solución de problemas](#)

Cómo funciona Amazon EMR en EKS con Lake Formation AWS

El uso de Amazon EMR en EKS con Lake Formation le permite aplicar una capa de permisos en cada trabajo de Spark para aplicar el control de permisos de Lake Formation cuando Amazon EMR en EKS ejecuta trabajos. Amazon EMR en EKS 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 aplica las políticas de Lake Formation. 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 más información, consulte [Qué es AWS Lake Formation](#).

La siguiente es una descripción general de alto nivel de cómo Amazon EMR en EKS obtiene acceso a los datos protegidos por las políticas de seguridad de Lake Formation.



Los siguientes pasos describen este proceso:

1. Un usuario envía un trabajo de Spark a un clúster virtual Amazon EMR en EKS habilitado para AWS Lake Formation.
2. El servicio Amazon EMR on EKS configura el controlador de usuario y ejecuta el trabajo 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 Amazon S3 ni al catálogo de datos de Glue. Solo crea un plan de trabajo.
3. El servicio Amazon EMR en EKS configura un segundo controlador denominado controlador del sistema y lo ejecuta en el perfil del sistema (con una identidad privilegiada). Amazon EKS 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. Funciona con Spark completo y se comunica con Amazon S3 y el catálogo de datos para acceder a los datos. Solicita ejecutores y compila el Job Plan en una secuencia de etapas de ejecución.

4. A continuación, Amazon EMR en el servicio EKS ejecuta las etapas en los ejecutores. El código de usuario en cualquier etapa 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 Lake Formation o las que aplican filtros de seguridad se delegan a los ejecutores del sistema.

Habilite Lake Formation con Amazon EMR en EKS

Con Amazon EMR versión 7.7 y versiones posteriores, puede aprovechar AWS Lake Formation para aplicar controles de acceso detallados en las tablas del catálogo de datos respaldadas por Amazon 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 su Amazon EMR en EKS Spark Jobs.

En esta sección se explica cómo crear una configuración de seguridad y cómo configurar Lake Formation para que funcione con Amazon EMR. También describe cómo crear un clúster virtual con la configuración de seguridad que creó para Lake Formation. Estas secciones deben completarse en secuencia.

Paso 1: Configure los permisos a nivel de columna, fila o celda basados en Lake Formation

En primer lugar, para aplicar permisos a nivel de fila y columna con Lake Formation, el administrador del lago de datos de Lake Formation debe establecer la etiqueta de `LakeFormationAuthorizedCallersesión`. Lake Formation usa esta etiqueta de sesión para autorizar a los intermediarios y proporcionar acceso al lago de datos.

Navegue a la consola de AWS Lake Formation y seleccione la opción de configuración de integración de aplicaciones en la sección Administración de la barra lateral. A continuación, active la casilla Permitir que motores externos filtren los datos en las ubicaciones de Amazon S3 registradas en Lake Formation. Añada la AWS cuenta IDs en la que se ejecutarían los trabajos de Spark y los valores de la etiqueta de sesión.

Application integration settings [Learn more](#)

Application integration settings

Use the options below to control which third-party engines are allowed to read and filter data in Amazon S3 locations registered with Lake Formation.

☒ **Allow external engines to filter data in Amazon S3 locations registered with Lake Formation**

Check this box to allow third-party engines to access data in Amazon S3 locations that are registered with Lake Formation.

Session tag values

Enter one or more strings that match the LakeFormationAuthorizedCaller session tag defined for third-party engines.

[Clear all](#)

EMR on EKS Engine ✕

Enter one or several string values separated by comma.

AWS account IDs

Enter the external AWS account IDs from where third-party engines are allowed to access locations registered with Lake Formation.

[Clear all](#)

012345678901 ✕

Account

Enter one or more AWS account IDs. Press enter after each ID.

☐ **Allow external engines to access data in Amazon S3 locations with full table access**

When you enable this option, Lake Formation will return credentials to the integrated application directly without IAM session tag validation.

[Cancel](#)[Save](#)

Tenga en cuenta que la etiqueta de LakeFormationAuthorizedCallersesión que se pasa aquí se incluye SecurityConfigurationmás adelante al configurar las funciones de IAM, en la sección 3.

Paso 2: Configurar los permisos RBAC de EKS

En segundo lugar, se configuran los permisos para el control de acceso basado en roles.

Proporcione permisos de clúster EKS a Amazon EMR en el servicio EKS

El servicio Amazon EMR on EKS debe tener permisos de rol de clúster de EKS para poder crear permisos de espacios de nombres cruzados para que el controlador del sistema genere ejecutores de usuarios en el espacio de nombres de usuario.

Cree un rol de clúster

En este ejemplo, se definen los permisos para un conjunto de recursos.

```
vim emr-containers-cluster-role.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: emr-containers
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods", "pods/
log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
"deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
"patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
"patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
"patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles", "clusterrolebindings", "roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
"deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
"deletecollection", "annotate", "patch", "label"]
---
```

```
kubectl apply -f emr-containers-cluster-role.yaml
```

Cree enlaces de roles de clúster

```
vim emr-containers-cluster-role-binding.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: emr-containers
subjects:
- kind: User
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
---
```

```
kubectl apply -f emr-containers-cluster-role-binding.yaml
```

Proporcionar acceso al espacio de nombres a Amazon EMR en el servicio EKS

Cree dos espacios de nombres de Kubernetes, uno para el controlador de usuario y los ejecutores y otro para el controlador y los ejecutores del sistema, y habilite Amazon EMR en el acceso al servicio EKS para enviar trabajos en los espacios de nombres de usuario y sistema. [Siga la guía existente para proporcionar acceso a cada espacio de nombres, que está disponible en *Habilitar el acceso al clúster mediante aws-auth*](#)

Paso 3: Configurar las funciones de IAM para los componentes del perfil de usuario y del sistema

En tercer lugar, debe configurar los roles para componentes específicos. Un Spark Job habilitado para Lake Formation tiene dos componentes: usuario y sistema. El controlador de usuario y los ejecutores se ejecutan en el espacio de nombres de usuario y están vinculados al JobExecutionRole que se transfiere a la API. StartJobRun El controlador y los ejecutores del sistema se ejecutan en el espacio de nombres del sistema y están vinculados a la función. QueryEngine

Configure la función de Query Engine

El QueryEngine rol está vinculado a los componentes del espacio del sistema y tendría permisos para asumir la etiqueta JobExecutionRolewith LakeFormationAuthorizedCallerSession. La política de permisos de IAM del rol de Query Engine es la siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeJobRoleWithSessionTagAccessForSystemDriver",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Resource": "arn:aws:iam::Account:role/JobExecutionRole",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS Engine"
        }
      }
    },
    {
      "Sid": "AssumeJobRoleWithSessionTagAccessForSystemExecutor",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::Account:role/JobExecutionRole",
    },
    {
      "Sid": "CreateCertificateAccessForTLS",
      "Effect": "Allow",
      "Action": "emr-containers:CreateCertificate",
      "Resource": "*"
    }
  ]
}
```

Configure la política de confianza del rol Query Engine para que confíe en el espacio de nombres del sistema Kubernetes.


```
aws emr-containers update-role-trust-policy \
  --cluster-name eks cluster \
  --namespace eks system namespace \
  --role-name query_engine_iam_role_name
```

Para obtener más información, consulte [Actualización](#) de la política de confianza de roles.

Configurar el rol de ejecución de tareas

Los permisos de Lake Formation controlan el acceso a los recursos del catálogo de datos de AWS Glue, a las ubicaciones de Amazon S3 y a los datos subyacentes en esas ubicaciones. Los permisos de la IAM controlan el acceso a Lake Formation and AWS Glue APIs y a los recursos. Aunque es posible que tenga el permiso de Lake Formation para acceder a una tabla del catálogo de datos (SELECT), su operación fallará si no tiene el permiso de IAM en las operaciones de la `glue:Get*` API.

Política de permisos de IAM de JobExecutionRole: el JobExecutionrol debe incluir las declaraciones de política en su política de permisos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": ["*"]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": ["*"]
    }
  ]
}
```

```

        "Sid": "CreateCertificateAccessForTLS",
        "Effect": "Allow",
        "Action": "emr-containers:CreateCertificate",
        "Resource": "*"
    }
]
}

```

Política de confianza de IAM para: JobExecutionRole

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TrustQueryEngineRoleForSystemDriver",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::your_account:role/QueryExecutionRole"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Condition": {
        "StringLike": {
          "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS Engine"
        }
      }
    },
    {
      "Sid": "TrustQueryEngineRoleForSystemExecutor",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::your_account:role/QueryEngineRole"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Configure el rol Trust Policy of Job Execution para que confíe en el espacio de nombres de usuario de Kubernetes:

```
aws emr-containers update-role-trust-policy \
  --cluster-name eks cluster \
  --namespace eks User namespace \
  --role-name job_execution_role_name
```

Para obtener más información, consulte [Actualizar la política de confianza del rol de ejecución de tareas](#).

Paso 4: Configurar la configuración de seguridad

Para ejecutar un trabajo habilitado para Lake Formation, debe crear una configuración de seguridad.

```
aws emr-containers create-security-configuration \
  --name 'security-configuration-name' \
  --security-configuration '{
    "authorizationConfiguration": {
      "lakeFormationConfiguration": {
        "authorizedSessionTagValue": "SessionTag configured in LakeFormation",
        "secureNamespaceInfo": {
          "clusterId": "eks-cluster-name",
          "namespace": "system-namespace-name"
        },
        "queryEngineRoleArn": "query-engine-IAM-role-ARN"
      }
    }
  }'
```

Asegúrese de que la etiqueta de sesión introducida en el campo `authorizedSessionTagValue` pueda autorizar a Lake Formation. Establezca el valor como el configurado en Lake Formation, in [Paso 1: Configure los permisos a nivel de columna, fila o celda basados en Lake Formation](#).

Paso 5: Crear un clúster virtual

Cree un clúster virtual de Amazon EMR en EKS con una configuración de seguridad.

```
aws emr-containers create-virtual-cluster \
  --name my-lf-enabled-vc \
  --container-provider '{
    "id": "eks-cluster",
    "type": "EKS",
    "info": {
      "eksInfo": {
```

```

        "namespace": "user-namespace"
    }
}
}' \
--security-configuration-id SecurityConfiguraionId

```

Asegúrese de pasar el SecurityConfigurationID del paso anterior para que la configuración de autorización de Lake Formation se aplique a todos los trabajos que se ejecutan en el clúster virtual. Para obtener más información, consulte [Registrar el clúster de Amazon EKS en Amazon EMR](#).

Paso 6: Enviar un trabajo en el FGAC Enabled VirtualCluster

El proceso de presentación de trabajos es el mismo para los trabajos que no son de Lake Formation y Lake Formation. Para obtener más información, consulte [Enviar un trabajo realizado con StartJobRun](#).

El controlador de Spark, el ejecutor y los registros de eventos del controlador del sistema se almacenan en el depósito S3 de la cuenta de AWS servicio para su depuración. Recomendamos configurar una clave KMS administrada por el cliente en Job Run para cifrar todos los registros almacenados en el AWS depósito de servicio. Para obtener más información sobre cómo habilitar el cifrado de registros, consulte [Cifrar Amazon EMR en](#) registros de EKS.

Consideraciones y limitaciones

Tenga en cuenta las siguientes consideraciones y limitaciones cuando utilice Lake Formation con Amazon EMR en EKS:

- Amazon EMR en EKS admite un control de acceso detallado a través de Lake Formation solo para los formatos de tabla Apache Hive, Apache Iceberg, Apache Hudi y Delta. Los formatos de Apache Hive incluyen Parquet, ORC y xSV.
- DynamicResourceAllocation está activado de forma predeterminada y no se puede desactivar DynamicResourceAllocation para los trabajos de Lake Formation. Como el valor predeterminado de la `spark.dynamicAllocation.maxExecutors` configuración DRA es infinito, configure un valor adecuado en función de su carga de trabajo.
- Los trabajos habilitados para Lake Formation no admiten el uso de EMR personalizados en las imágenes de EKS en los controladores y ejecutores del sistema.
- Solo puede utilizar Lake Formation con trabajos de Spark.
- El EMR en EKS con Lake Formation solo admite una sesión de Spark durante un trabajo.

- EMR en EKS 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
- El EMR en EKS bloquea las funcionalidades que podrían socavar el aislamiento total del controlador del sistema, incluidas las siguientes:
 - UDTs, Hive UDFs y cualquier función definida por el usuario que incluya clases personalizadas
 - Orígenes de datos personalizados
 - Suministro de tarros adicionales para la extensión, el conector o el comando metastore de Spark
ANALYZE TABLE
- Para hacer cumplir los controles de acceso, EXPLAIN PLAN y las operaciones de DDL, como DESCRIBE TABLE, no exponen información restringida.
- Amazon EMR en EKS restringe el acceso a los registros de Spark del controlador del sistema en los trabajos habilitados para la formación de lagos. Dado que el controlador del sistema se ejecuta con más acceso, los eventos y registros que genera el controlador del sistema pueden incluir información confidencial. Para evitar que usuarios o códigos no autorizados accedan a estos datos confidenciales, EMR en EKS deshabilitó el acceso a los registros de los controladores del sistema. Para solucionar problemas, póngase en contacto con el servicio de AWS asistencia.
- 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 la función de ejecución de tareas de EMR en EKS. Si configura mal el rol registrado con la ubicación de la tabla, se producirá un error en los trabajos enviados que utilicen el rol con el permiso de S3 IAM para acceder a la ubicación de la tabla.
- Para escribir en una tabla de Lake Formation se utiliza el permiso de IAM en lugar de los permisos concedidos por Lake Formation. Si su función de ejecución de trabajos tiene los permisos de S3 necesarios, puede utilizarla para ejecutar operaciones de escritura.

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.
- Le recomendamos que utilice Iceberg `DataFrameWriter V2` en lugar de `V1`.

Para obtener más información, consulte [Comprender los conceptos y la terminología de Amazon EMR en EKS](#) y [Habilitar el acceso a clústeres para Amazon EMR](#) en EKS.

Exención de responsabilidad para los administradores de datos

Note

Al conceder acceso a los recursos de Lake Formation a una función de IAM para EMR en EKS, debe asegurarse de que el administrador u operador del clúster de EMR sea un administrador de confianza. Esto es particularmente relevante para los recursos de Lake Formation que se comparten entre varias organizaciones y AWS cuentas.

Responsabilidades de los administradores de EKS

- El espacio de `System` nombres debe estar protegido. No se permitirá que ningún usuario, recurso, entidad o herramienta tenga permisos RBAC de Kubernetes en los recursos de Kubernetes del espacio de nombres. `System`
- Ningún usuario, recurso o entidad, excepto el EMR en el servicio EKS, debe tener `CREATE` acceso a `POD`, `CONFIG_MAP` y `SECRET` en el espacio de nombres. `User`
- `System` los controladores y ejecutores contienen datos confidenciales. `System` Por lo tanto, los eventos de Spark, los registros de los controladores de Spark y los registros de los ejecutores de Spark del espacio de `System` nombres no deberían reenviarse a sistemas de almacenamiento de registros externos.

Solución de problemas

Registro

EMR en EKS utiliza los perfiles de recursos de Spark para dividir la ejecución de los trabajos. Amazon EMR en EKS utiliza el perfil de usuario para ejecutar el código que ha proporcionado, mientras que el perfil del sistema aplica las políticas de Lake Formation. Puede acceder a los registros de los contenedores ejecutados como perfil de usuario configurando la StartJobRun solicitud con. [MonitoringConfiguration](#)

Servidor de historial de Spark

El servidor de historial de Spark contiene todos los eventos de Spark generados a partir del perfil de usuario y los eventos redactados generados a partir del controlador del sistema. Puedes ver todos los contenedores de los controladores de usuario y del sistema en la pestaña Ejecutores. Sin embargo, los enlaces de registro solo están disponibles para el perfil de usuario.

Error de trabajo: permisos de Lake Formation insuficientes

Asegúrese de que su función de ejecución de tareas tenga los permisos para ejecutarse SELECT y esté DESCRIBE en la tabla a la que está accediendo.

Error de trabajo en la ejecución de RDD

Actualmente, el EMR en EKS no admite operaciones de conjuntos de datos distribuidos resilientes (RDD) en trabajos habilitados para la formación de lagos.

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

El EMR de EKS detectó un error de validación de seguridad. Póngase en contacto con AWS el servicio de asistencia para obtener ayuda

Compartir el catálogo de datos y las tablas de AWS Glue entre cuentas

Puede compartir bases de datos y tablas entre cuentas y seguir utilizando Lake Formation. Para obtener más información, consulte [Intercambio de datos entre cuentas en Lake Formation](#) y [¿Cómo](#)

[comparto el catálogo de datos y las tablas de AWS Glue entre cuentas mediante AWS Lake](#)

Formation? .

Iceberg Job lanza un error de inicialización que no configura la región AWS

El mensaje es el siguiente:

```
25/02/25 13:33:19 ERROR SparkFGACEExceptionSanitizer: Client received error with id =  
b921f9e6-f655-491f-b8bd-b2842cdc20c7,  
reason = IllegalArgumentException, message = Cannot initialize  
LakeFormationAwsClientFactory, please set client.region to a valid aws region
```

Asegúrese de que la configuración de Spark

`spark.sql.catalog.catalog_name.client.region` esté establecida en una región válida.

Lanzamiento de Iceberg Job SparkUnsupportedOperationException

El mensaje es el siguiente:

```
25/02/25 13:53:15 ERROR SparkFGACEExceptionSanitizer: Client received error with id =  
921fef42-0800-448b-bef5-d283d1278ce0,  
reason = SparkUnsupportedOperationException, message = Either glue.id or glue.account-  
id is set with non-default account.  
Cross account access with fine-grained access control is only supported with AWS  
Resource Access Manager.
```

Asegúrese de que la configuración de `spark.sql.catalog.catalog_name.glue.account-id` Spark tenga un identificador de cuenta válido.

Registro y supervisión

Para detectar incidentes, recibir alertas cuando ocurran y responder a ellas, utilice estas opciones con Amazon EMR en EKS:

- Supervise Amazon EMR en EKS con AWS CloudTrail - [AWS CloudTrail](#) proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Amazon EMR en EKS. Captura las llamadas desde la consola de Amazon EMR y las llamadas de código a las operaciones de la API de Amazon EMR en EKS como eventos. Esto le permite determinar la solicitud que se envió a Amazon EMR en EKS, la dirección IP desde la que se hizo la solicitud,

quién la hizo, cuándo se hizo y detalles adicionales. Para obtener más información, consulte [Registro de Amazon EMR en las llamadas a la API de EKS mediante AWS CloudTrail](#).

- Use CloudWatch Events con Amazon EMR en EKS - CloudWatch Events ofrece una transmisión casi en tiempo real de los eventos del sistema que describen los cambios en AWS los recursos. CloudWatch Events se percata de los cambios operativos a medida que se producen, responde a ellos y toma las medidas correctivas necesarias mediante el envío de mensajes en respuesta al entorno, la activación de funciones, la realización de cambios y la captura de información de estado. Para usar CloudWatch Events with Amazon EMR en EKS, cree una regla que se active en una llamada a la API de Amazon EMR on EKS mediante. CloudTrail Para obtener más información, consulte [Supervisa los trabajos con Amazon CloudWatch Events](#).

Cifrado de Amazon EMR en registros de EKS con almacenamiento gestionado

En las siguientes secciones, se muestra cómo configurar el cifrado de los registros.

Enable encryption

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

```
"monitoringConfiguration": {
  "managedLogs": {
    "allowAWSToRetainLogs": "ENABLED",
    "encryptionKeyArn": "KMS key arn"
  },
  "persistentAppUI": "ENABLED"
}
```

La `allowAWSToRetainLogs` configuración permite AWS conservar los registros del espacio de nombres del sistema cuando se ejecuta un trabajo con el FGAC nativo. La `persistentAppUI` configuración permite guardar los registros AWS de eventos que se utilizan para generar la interfaz de usuario de Spark. `encryptionKeyArn` se usa para especificar el ARN de clave KMS que desea usar para cifrar los registros almacenados en AWS.

Permisos necesarios para el cifrado de registros

El usuario que envíe el trabajo o vea la interfaz de usuario de Spark debe poder realizar las acciones `kms:DescribeKey` y `kms:Decrypt` obtener la clave de cifrado. `kms:GenerateDataKey` Estos

permisos se utilizan para verificar la validez de la clave y comprobar que el usuario tiene los permisos necesarios para leer y escribir los registros cifrados con la clave KMS. Si el usuario que envía el trabajo carece de los permisos clave necesarios, Amazon EMR en EKS rechaza el envío de la ejecución del trabajo.

Ejemplo de política de IAM para el rol utilizado para llamar StartJobRun

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "emr-containers:StartJobRun",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:DescribeKey",
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "KMS key ARN",
      "Effect": "Allow"
    }
  ]
}
```

También debe configurar la clave KMS para permitir que los directores `elasticmapreduce.amazonaws.com` de servicio puedan `persistentappui.elasticmapreduce.amazonaws.com` y `kms:GenerateDataKey` `kms:Decrypt`. Esto permite a EMR leer y escribir registros cifrados con la clave KMS en el almacenamiento gestionado.

Ejemplo de política de claves de KMS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "IAM role ARN used to call StartJobRun"
      }
    }
  ]
}
```

```

    },
    "Action": "kms:DescribeKey",
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "kms:viaService": "emr-containers.region.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "IAM role ARN used to call StartJobRun"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "kms:viaService": "emr-containers.region.amazonaws.com",
            "kms:EncryptionContext:aws:emr-containers:virtualClusterId":
"virtual cluster id"
        }
    }
},
{
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "persistentappui.elasticmapreduce.amazonaws.com",
            "elasticmapreduce.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "kms:EncryptionContext:aws:emr-containers:virtualClusterId":
"virtual cluster id",

```

```

    "aws:SourceArn": "virtual cluster ARN"
  }
}
]
}

```

Como práctica recomendada de seguridad, le recomendamos que añada las `aws:SourceArn` condiciones `kms:viaServicekms:EncryptionContext`, y. Estas condiciones ayudan a garantizar que Amazon EMR solo utilice la clave en EKS y solo se utilice para los registros generados a partir de trabajos que se ejecutan en un clúster virtual específico.

Registro de Amazon EMR en las llamadas a la API de EKS mediante AWS CloudTrail

Amazon EMR en EKS está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Amazon EMR en EKS. CloudTrail captura todas las llamadas a la API de Amazon EMR en EKS como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de Amazon EMR en EKS y las llamadas desde el código a las operaciones de la API de Amazon EMR en EKS. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Amazon EMR en EKS. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon EMR en EKS, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la [Guía del AWS CloudTrail usuario](#).

Información sobre Amazon EMR sobre EKS en CloudTrail

CloudTrail está activado en su AWS cuenta al crear la cuenta. Cuando se produce actividad en Amazon EMR en EKS, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su AWS cuenta. Para obtener más información, consulta [Cómo ver eventos con el historial de CloudTrail eventos](#).

Para obtener un registro continuo de los eventos de su AWS cuenta, incluidos los eventos de Amazon EMR en EKS, cree un registro. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando crea una ruta en la consola, la ruta

se aplica a todas AWS las regiones. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- [Introducción a la creación de registros de seguimiento](#)
- [CloudTrail servicios e integraciones compatibles](#)
- [Configuración de las notificaciones de Amazon SNS para CloudTrail](#)
- [Recibir archivos de CloudTrail registro de varias regiones](#) y [recibir archivos de CloudTrail registro de varias cuentas](#)

Todas las acciones de Amazon EMR en EKS se registran CloudTrail y documentan en la documentación de la API de Amazon [EMR en EKS](#). Por ejemplo, las llamadas a `StartJobRun` y `ListJobRuns` las acciones generan entradas en los archivos de CloudTrail registro. `CreateVirtualCluster`

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o AWS Identity and Access Management (IAM).
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el [elemento CloudTrail de identidad del usuario](#).

Descripción de las entradas de archivos de registro de Amazon EMR en EKS

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la [ListJobRuns](#) acción.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-04T21:49:36Z"
      }
    }
  },
  "eventTime": "2020-11-04T21:52:58Z",
  "eventSource": "emr-containers.amazonaws.com",
  "eventName": "ListJobRuns",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "virtualClusterId": "1K48XXXXXXHCB"
  },
  "responseElements": null,
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```

Uso de Amazon S3 Access Grants con Amazon EMR en EKS

Información general de S3 Access Grants para Amazon EMR en EKS

A partir de la versión 6.15.0 de Amazon EMR, Amazon S3 Access Grants proporcionan una solución de control de acceso escalable que puede utilizar para aumentar el acceso a los datos de Amazon S3 desde Amazon EMR en EKS. Si cuenta con una configuración de permisos compleja o amplia de datos de S3, puede utilizar Access Grants para escalar los permisos de datos de S3 para usuarios, roles y aplicaciones.

Utilice S3 Access Grants para 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 en EKS.

Para obtener más información, consulte [Administración del acceso con S3 Access Grants para 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.

Esta página, se describen los requisitos para ejecutar un trabajo de Spark en Amazon EMR en EKS con la integración de S3 Access Grants. Con Amazon EMR en EKS, S3 Access Grants requiere una instrucción de política de IAM adicional en la función de ejecución de su trabajo y una configuración de anulación adicional para la API `StartJobRun`. Para conocer los pasos para configurar 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 EMR sin servidor](#)

Lanzamiento de un clúster de Amazon EMR en EKS con S3 Access Grants para la administración de datos

Puede habilitar S3 Access Grants en Amazon EMR en EKS y ejecutar un trabajo de 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 clúster de Amazon EMR en EKS. Incluya los permisos de IAM necesarios para ejecutar los trabajos de Spark, `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, es posible que los usuarios puedan acceder a más datos además de los que usted define en S3 Access Grants

- Envíe un trabajo a su clúster de Amazon EMR en EKS con una etiqueta de versión de Amazon EMR de 6.15 o superior y la clasificación `emrfs-site`, como se muestra en el siguiente ejemplo. Reemplace los valores en *red text* con valores adecuados para su caso de uso.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-7.8.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2"],
      "sparkSubmitParameters": "--class main_class"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emrfs-site",
        "properties": {
          "fs.s3.s3AccessGrants.enabled": "true",

```



```
        "fs.s3.s3AccessGrants.fallbackToIAM": "false"  
      }  
    },  
    ],  
  }  
}
```

Consideraciones sobre el uso de S3 Access Grants con Amazon EMR en EKS

Para obtener información importante sobre soporte, compatibilidad y comportamiento al utilizar Amazon S3 Access Grants con Amazon EMR en EKS, consulte [Consideraciones sobre el uso de S3 Access Grants con Amazon EMR](#) en la Guía de administración de Amazon EMR.

Validación de la conformidad de Amazon EMR en EKS

Los auditores externos evalúan la seguridad y el cumplimiento de Amazon EMR en EKS como parte de varios programas de AWS cumplimiento. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros.

Resiliencia de Amazon EMR en EKS

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

[Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS](#)

Además de la infraestructura AWS global, Amazon EMR en EKS ofrece integración con Amazon S3 a través de EMRFS para ayudarlo a satisfacer sus necesidades de respaldo y resiliencia de datos.

Seguridad de la infraestructura de Amazon EMR en EKS

Como servicio gestionado, Amazon EMR está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte [Seguridad AWS en la nube](#). Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte [Protección de infraestructuras en un marco](#) de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas 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.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puedes utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Configuración y análisis de vulnerabilidades

AWS gestiona las tareas de seguridad básicas, como la aplicación de parches al sistema operativo (SO) huésped y a las bases de datos, la configuración del firewall y la recuperación ante desastres. Estos procedimientos han sido revisados y certificados por los terceros pertinentes. Para obtener más detalles, consulte los siguientes recursos de :

- [Validación de la conformidad de Amazon EMR en EKS](#)
- [Modelo de responsabilidad compartida](#)
- [Amazon Web Services: información general de procesos de seguridad](#) (documento técnico)

Conexión a Amazon EMR en EKS mediante un punto de conexión de VPC de interfaz

Puede conectarse directamente a Amazon EMR en EKS mediante los [puntos de enlace de Interface VPC \(\) de su Virtual Private Cloud \(VPC AWS PrivateLink\)](#) en lugar de conectarse a través de Internet. Cuando utiliza un punto final de VPC de interfaz, la comunicación entre su VPC y Amazon EMR en EKS se lleva a cabo íntegramente dentro de la red. AWS Cada punto final de la VPC está representado por una o más [interfaces de red elásticas \(ENIs\)](#) con direcciones IP privadas en las subredes de la VPC.

El punto de conexión de la VPC de la interfaz conecta la VPC directamente a Amazon EMR en EKS sin necesidad de una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión Direct Connect. AWS Las instancias de la VPC no necesitan direcciones IP públicas para comunicarse con la API de Amazon EMR en EKS.

Puede crear un punto de enlace de VPC de interfaz para conectarse a Amazon EMR en EKS mediante los comandos AWS Management Console o AWS Command Line Interface ().AWS CLI Para obtener más información, consulte [Creación de un punto de conexión de interfaz](#).

Después de crear un punto de conexión de VPC de interfaz, si habilita nombres de host de DNS privados para el punto de conexión, el punto de conexión predeterminado de Amazon EMR en EKS se resuelve en el punto de conexión de VPC. El punto de conexión del nombre de servicio predeterminado para Amazon EMR en EKS tiene el siguiente formato.

```
emr-containers.Region.amazonaws.com
```

Si no habilita nombres de host de DNS privados, Amazon VPC proporciona un nombre de punto de conexión de DNS que puede utilizar en el siguiente formato.

```
VPC_Endpoint_ID.emr-containers.Region.vpce.amazonaws.com
```

Para obtener más información, consulte [Interface VPC Endpoints \(AWS PrivateLink\)](#) en la Guía del usuario de Amazon VPC. Amazon EMR en EKS permite llamar a todas sus [Acciones de la API](#) en su VPC.

Puede adjuntar políticas de punto de conexión de VPC a un punto de conexión de VPC para controlar el acceso de las entidades principales de IAM. También puede asociar grupos de seguridad

con un punto de conexión de VPC para controlar el acceso de entrada y salida en función del origen y el destino del tráfico de red, como un rango de direcciones IP. Para obtener más información, consulte [Control del acceso a los servicios con Puntos de conexión de VPC](#).

Crear una política de punto de conexión de VPC para Amazon EMR en EKS

Puede crear una política para los puntos de conexión de VPC de Amazon para Amazon EMR en EKS y especificar lo siguiente:

- La entidad principal que puede o no puede realizar acciones
- Las acciones que se pueden realizar
- Los recursos en los que se pueden llevar a cabo las acciones

Para obtener más información, consulte [Controlar el acceso a servicios con puntos de conexión de VPC](#) en la Guía del usuario de Amazon VPC.

Example Política de punto final de VPC para denegar todo el acceso desde una cuenta específica AWS

La siguiente política de punto final de VPC deniega a la AWS cuenta **123456789012** todo acceso a los recursos que utilizan el punto final.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": {
        "AWS": [
          "123456789012"
        ]
      }
    }
  ]
}
```

```

    }
  ]
}
```

Example Política de punto de conexión de VPC para permitir el acceso de VPC solo a una entidad principal de IAM especificada (usuario)

La siguiente política de puntos finales de VPC permite el acceso total solo al usuario *Lijuan* de IAM de la cuenta. AWS *123456789012* A las demás entidades principales de IAM se les deniega el acceso a través del punto de enlace.

```

{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/Lijuan"
        ]
      }
    }
  ]
}
```

Example Política de punto de conexión de VPC para permitir operaciones de Amazon EMR en EKS de solo lectura

La siguiente política de puntos finales de VPC permite que solo la AWS cuenta realice *123456789012* las acciones de Amazon EMR especificadas en EKS.

Las acciones especificadas proporcionan el equivalente al acceso de solo lectura para Amazon EMR en EKS. Las demás acciones en la VPC se deniegan para la cuenta especificada. A las demás cuentas se les deniega el acceso. Para obtener una lista de acciones de Amazon EMR en EKS, consulte [Acciones, recursos y claves de condición de Amazon EMR en EKS](#).

```

{
  "Statement": [
    {
      "Action": [
```

```

        "emr-containers:DescribeJobRun",
        "emr-containers:DescribeVirtualCluster",
        "emr-containers:ListJobRuns",
        "emr-containers:ListTagsForResource",
        "emr-containers:ListVirtualClusters"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Principal": {
        "AWS": [
            "123456789012"
        ]
    }
}
]
}

```

Example Política de punto de conexión de VPC que deniega el acceso a un clúster virtual específico

La siguiente política de puntos finales de VPC permite el acceso total a todas las cuentas y entidades principales, pero deniega el acceso de la AWS cuenta **123456789012** a las acciones realizadas en el clúster virtual con el ID de clúster. **A1B2CD34EF5G** Se siguen permitiendo otras acciones de Amazon EMR en EKS que no admiten permisos en el nivel de los recursos para los clústeres virtuales. Para obtener una lista de las acciones de Amazon EMR en EKS y los tipos de recursos correspondientes, consulte [Acciones, recursos y claves de condición de Amazon EMR en EKS](#) en la Guía del usuario de AWS Identity and Access Management .

```

{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "arn:aws:emr-containers:us-west-2:123456789012:/virtualclusters/A1B2CD34EF5G",
      "Principal": {
        "AWS": [

```

```
    "123456789012"  
  ]  
}  
]  
}
```

Configuración del acceso entre cuentas de Amazon EMR en EKS

Puede configurar el acceso entre cuentas de Amazon EMR en EKS. El acceso entre cuentas permite a los usuarios de una AWS cuenta ejecutar Amazon EMR en trabajos de EKS y acceder a los datos subyacentes que pertenecen a AWS otra cuenta.

Requisitos previos

Para configurar el acceso multicuenta a Amazon EMR en EKS, deberá completar las tareas mientras haya iniciado sesión en las AWS siguientes cuentas:

- **AccountA-** Una AWS cuenta en la que ha creado un Amazon EMR en un clúster virtual de EKS mediante el registro de Amazon EMR con un espacio de nombres de un clúster de EKS.
- **AccountB-** Una AWS cuenta que contenga un bucket de Amazon S3 o una tabla de DynamoDB a la que desee que accedan sus trabajos de Amazon EMR en EKS.

Debe tener lo siguiente en sus AWS cuentas antes de configurar el acceso entre cuentas:

- Un clúster virtual de Amazon EMR en EKS en la AccountA donde desee ejecutar los trabajos.
- Un rol de ejecución de trabajos de la AccountA que tiene los permisos necesarios para ejecutar trabajos en el clúster virtual. Para obtener más información, consulte [Crear un rol de ejecución de trabajos](#) y [Uso de roles de ejecución de trabajos con Amazon EMR en EKS](#).

Cómo acceder a un bucket de Amazon S3 en diversas cuentas o a una tabla de DynamoDB

Para configurar el acceso entre cuentas de Amazon EMR en EKS, complete los siguientes pasos.

1. Cree un bucket de Amazon S3, `cross-account-bucket`, en la AccountB. Para obtener más información, consulte [Creación de un bucket](#). 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](#).

2. Cree un rol de IAM Cross-Account-Role-B en la AccountB que pueda acceder a cross-account-bucket.
 1. Inicie sesión en la consola de IAM.
 2. 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.
 3. Cree una política de IAM que especifique los permisos del Cross-Account-Role-B para acceder al bucket de S3 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 una política nueva](#) en la Guía del usuario de IAM.

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

Si se 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 [Creación de una tabla de DynamoDB](#) en la Guía del usuario de IAM.

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

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



```

    {
      "Effect": "Allow",
      "Action": "dynamodb:*",
      "Resource": "arn:aws:dynamodb:MyRegion:AccountB:table/
CrossAccountTable"
    }
  ]
}

```

3. Edite la relación de confianza del rol Cross-Account-Role-B.

1. Para configurar la relación de confianza del rol, elija la pestaña Relaciones de confianza en la consola de IAM para el rol creado en el paso 2: Cross-Account-Role-B.
2. Seleccione Editar la relación de confianza.
3. Agregue el siguiente documento de política, que permite al Job-Execution-Role-A de la AccountA asumir este rol Cross-Account-Role-B.

```

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

```

4. Otorgue al Job-Execution-Role-A de la AccountA el permiso de asunción de roles de STS para asumir el rol Cross-Account-Role-B.

1. En la consola de IAM de la AWS cuentaAccountA, selecciona. Job-Execution-Role-A
2. Agregue la siguiente instrucción de política al Job-Execution-Role-A para denegar la acción AssumeRole en el rol Cross-Account-Role-B.

```

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

```

```

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

```

5. Para acceder a Amazon S3, defina los siguientes parámetros `spark-submit` (`spark conf`) al enviar el trabajo a Amazon EMR en EKS.

Note

De forma predeterminada, EMRFS usa el rol de ejecución del trabajo para acceder al bucket de S3 desde el trabajo. Sin embargo, cuando `customAWSCredentialsProvider` se establece en `AssumeRoleAWSCredentialsProvider`, EMRFS utiliza el rol correspondiente que especifique con `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` en lugar del `Job-Execution-Role-A` para el acceso a Amazon S3.

- `--conf`
`spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentialsProvider`
- `--conf`
`spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/Cross-Account-Role-B \`
- `--conf`
`spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/Cross-Account-Role-B \`

Note

Debe configurar `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` para el env tanto de ejecutor como controlador en la configuración de trabajos de Spark.

Para el acceso entre cuentas de DynamoDB, debe configurar `--conf`
`spark.dynamodb.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentialsProvider`

6. Ejecute el trabajo de Amazon EMR en EKS con el acceso entre cuentas, tal como se muestra en el siguiente ejemplo.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
"entryPointArguments": ["arguments_list"], "sparkSubmitParameters": "--class
<main_class> --conf spark.executor.instances=2 --conf spark.executor.memory=2G
--conf spark.executor.cores=2 --conf spark.driver.cores=1 --conf
spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
--conf
spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B --conf
spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B"}} ' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
"spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
"monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
{"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
"persistentAppUI": "ENABLED", "s3MonitoringConfiguration": {"logUri": "s3://
my_s3_log_location" }}}
```

Etiquetado de sus recursos de Amazon EMR en EKS

Para ayudarle a administrar los recursos de Amazon EMR en EKS, puede asignar sus propios metadatos a cada recurso con etiquetas. En este tema, se proporciona información general sobre la función de etiquetas y se muestra cómo puede crear etiquetas.

Temas

- [Conceptos básicos de etiquetas](#)
- [Etiquetar los recursos](#)
- [Restricciones de las etiquetas](#)
- [AWS CLI Trabaje con etiquetas mediante la API Amazon EMR en EKS](#)

Conceptos básicos de etiquetas

Una etiqueta es una etiqueta que se asigna a un AWS recurso. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario.

Las etiquetas te permiten clasificar AWS los recursos por atributos, como el propósito, el propietario o el entorno. Cuando tenga muchos recursos del mismo tipo, puede identificar rápidamente un recurso específico en función de las etiquetas que le haya asignado. Por ejemplo, puede definir un conjunto de etiquetas para los clústeres de Amazon EMR en EKS a fin de ayudar a hacer un seguimiento del propietario y del nivel de pila de cada clúster. Le recomendamos que diseñe un conjunto coherente de claves de etiqueta para cada tipo de recurso. Puede buscar y filtrar los recursos en función de las etiquetas que agregue.

Además, las etiquetas no se asignan a los recursos automáticamente. Después de agregar una etiqueta, puede editar las claves y los valores de las etiquetas o eliminar etiquetas de un recurso en cualquier momento. Si elimina un recurso, también se eliminará cualquier etiqueta asignada a dicho recurso.

Las etiquetas no tienen ningún significado semántico para Amazon EMR en EKS, por lo que se interpretan estrictamente como cadenas de caracteres.

Un valor de etiqueta puede ser una cadena vacía, pero no nulo. Una clave de etiqueta no puede ser una cadena vacía. Si agrega una etiqueta con la misma clave que una etiqueta existente en ese recurso, el nuevo valor sobrescribirá al anterior.

Si utilizas AWS Identity and Access Management (IAM), puedes controlar qué usuarios de tu AWS cuenta tienen permiso para gestionar las etiquetas.

Para ver ejemplos de políticas de control de acceso basadas en etiquetas, consulte [Políticas para el control de acceso basado en etiquetas](#).

Etiquetar los recursos

Puede etiquetar clústeres virtuales y ejecuciones de trabajos nuevos o existentes que se encuentren en estados activos. Los estados activos de las ejecuciones de tareas incluyen: PENDING, SUBMITTED, RUNNING y CANCEL_PENDING. Los estados activos de los clústeres virtuales incluyen: RUNNING, TERMINATING y ARRESTED. Para obtener más información, consulte [Estados de ejecuciones de trabajos](#) y [Estados del clúster virtual](#).

Cuando se finaliza un clúster virtual, las etiquetas se limpian y ya no se puede acceder a ellas.

Si utiliza Amazon EMR en la API de EKS, el o un AWS SDK AWS CLI, puede aplicar etiquetas a los nuevos recursos mediante el parámetro tags de la acción de API correspondiente. 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 en EKS que admiten etiquetas.

Recurso	Admite etiquetas	Admite la propagación de etiquetas	Admite el etiquetado o durante la creación (Amazon EMR en la API de EKS y el AWS CLI SDK) AWS	API para creación (se pueden agregar etiquetas durante la creación)
Clúster virtual	Sí	No. Las etiquetas	Sí	CreateVirtualCluster

Recurso	Admite etiquetas	Admite la propagación de etiquetas	Admite el etiquetado durante la creación (Amazon EMR en la API de EKS y el AWS CLI SDK) AWS	API para creación (se pueden agregar etiquetas durante la creación)
		asociadas a un clúster virtual no se propagan a las ejecuciones de trabajo enviadas a ese clúster virtual.		
Ejecuciones de trabajo	Sí	No	Sí	StartJobRun

Restricciones de las etiquetas

Se aplican las siguientes restricciones básicas a las etiquetas:

- Número máximo de etiquetas por recurso: 50
- Para cada recurso, cada clave de etiqueta debe ser única y solo puede tener un valor.
- Longitud máxima de la clave: 128 caracteres Unicode en UTF-8
- Longitud máxima del valor: 256 caracteres Unicode en UTF-8
- Si su esquema de etiquetado se utiliza en varios AWS servicios y recursos, recuerde que otros servicios pueden tener restricciones en cuanto a los caracteres permitidos. Los caracteres permitidos generalmente son: letras, números y espacios representables en UTF-8, además de los siguientes caracteres: + - = . _ : / @.
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.
- Un valor de etiqueta puede ser una cadena vacía, pero no nulo. Una clave de etiqueta no puede ser una cadena vacía.

- No utilice `aws :`, `AWS :`, ni ninguna combinación de mayúsculas o minúsculas del mismo como prefijo para claves o valores. Están reservados solo para su AWS uso.

AWS CLI Trabaje con etiquetas mediante la API Amazon EMR en EKS

Utilice los siguientes AWS CLI comandos o Amazon EMR en las operaciones de la API de EKS para añadir, actualizar, enumerar y eliminar las etiquetas de sus recursos.

Tarea	AWS CLI	Acción de la API
Agregar o sobrescribir una o varias etiquetas.	tag-resource	TagResource
Enumerar 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.

Ejemplo 1: etiquetar un clúster virtual existente

El siguiente comando etiqueta un clúster virtual existente.

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

Ejemplo 2: quitar la etiqueta de un clúster virtual existente

El siguiente comando elimina una etiqueta de un clúster virtual existente.

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

Ejemplo 3: enumerar etiquetas de un recurso

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

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


Solución de problemas de Amazon EMR en EKS

En esta sección, se describe cómo solucionar problemas con Amazon EMR en EKS. Para obtener información sobre cómo solucionar problemas generales con Amazon EMR, consulte [Solución de problemas de un clúster](#) en la Guía de administración de Amazon EMR.

Temas

- [Solución de problemas en trabajos que utilizan PersistentVolumeClaims \(PVC\)](#)
- [Solución de problemas en el escalado automático vertical de Amazon EMR en EKS](#)
- [Solución de problemas del operador de Spark de Amazon EMR en EKS](#)

Solución de problemas en trabajos que utilizan PersistentVolumeClaims (PVC)

Si necesitas crear, enumerar o eliminar PersistentVolumeClaims (PVC) para un trabajo, pero no añades permisos de PVC al rol `emr-containers` predeterminado de Kubernetes, el trabajo fallará cuando lo envíes. Sin estos permisos, el rol de `emr-containers` no puede crear los roles necesarios para el controlador o el cliente de Spark. No basta con agregar permisos a los roles de controlador o cliente de Spark, como sugieren los mensajes de error. El rol principal de `emr-containers` también debe incluir los permisos necesarios. En esta sección, se explica cómo agregar los permisos necesarios al rol principal de `emr-containers`.

Verificación

Para comprobar si su rol de `emr-containers` tiene o no los permisos necesarios, defina la variable `NAMESPACE` con su propio valor y, a continuación, ejecute el siguiente comando:

```
export NAMESPACE=YOUR_VALUE
kubectl describe role emr-containers -n ${NAMESPACE}
```

Además, para comprobar si los roles de Spark y de cliente tienen los permisos necesarios, ejecute el siguiente comando:

```
kubectl describe role emr-containers-role-spark-driver -n ${NAMESPACE}
kubectl describe role emr-containers-role-spark-client -n ${NAMESPACE}
```

Si los permisos no están disponibles, continúe con el parche de la siguiente manera.

Parche

1. Si los trabajos sin permisos se están ejecutando actualmente, deténgalos.
2. Cree un archivo denominado RBAC_Patch.py de la siguiente manera:

```
import os
import subprocess as sp
import tempfile as temp
import json
import argparse
import uuid

def delete_if_exists(dictionary: dict, key: str):
    if dictionary.get(key, None) is not None:
        del dictionary[key]

def doTerminalCmd(cmd):
    with temp.TemporaryFile() as f:
        process = sp.Popen(cmd, stdout=f, stderr=f)
        process.wait()
        f.seek(0)
        msg = f.read().decode()
    return msg

def patchRole(roleName, namespace, extraRules, skipConfirmation=False):
    cmd = f"kubectl get role {roleName} -n {namespace} --output json".split(" ")
    msg = doTerminalCmd(cmd)
    if "(NotFound)" in msg and "Error" in msg:
        print(msg)
        return False
    role = json.loads(msg)
    rules = role["rules"]
    rulesToAssign = extraRules[::]
    passedRules = []
    for rule in rules:
        apiGroups = set(rule["apiGroups"])
        resources = set(rule["resources"])
        verbs = set(rule["verbs"])
        for extraRule in extraRules:
            passes = 0
            apiGroupsExtra = set(extraRule["apiGroups"])
```

```

        resourcesExtra = set(extraRule["resources"])
        verbsExtra = set(extraRule["verbs"])
        passes += len(apiGroupsExtra.intersection(apiGroups)) >=
len(apiGroupsExtra)
        passes += len(resourcesExtra.intersection(resources)) >=
len(resourcesExtra)
        passes += len(verbsExtra.intersection(verbs)) >= len(verbsExtra)
        if passes >= 3:
            if extraRule not in passedRules:
                passedRules.append(extraRule)
                if extraRule in rulesToAssign:
                    rulesToAssign.remove(extraRule)
            break
    prompt_text = "Apply Changes?"
    if len(rulesToAssign) == 0:
        print(f"The role {roleName} seems to already have the necessary
permissions!")
        prompt_text = "Proceed anyways?"
    for ruleToAssign in rulesToAssign:
        role["rules"].append(ruleToAssign)
    delete_if_exists(role, "creationTimestamp")
    delete_if_exists(role, "resourceVersion")
    delete_if_exists(role, "uid")
    new_role = json.dumps(role, indent=3)
    uid = uuid.uuid4()
    filename = f"Role-{roleName}-New_Permissions-{uid}-TemporaryFile.json"
    try:
        with open(filename, "w+") as f:
            f.write(new_role)
            f.flush()
        prompt = "y"
        if not skipConfirmation:
            prompt = input(
                doTerminalCmd(f"kubect1 diff -f {filename}".split(" ")) +
f"\n{prompt_text} y/n: "
            ).lower().strip()
            while prompt != "y" and prompt != "n":
                prompt = input("Please make a valid selection. y/n:
").lower().strip()
            if prompt == "y":
                print(doTerminalCmd(f"kubect1 apply -f {filename}".split(" ")))
    except Exception as e:
        print(e)
    os.remove(f"./{filename}")

```

```

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("-n", "--namespace",
                        help="Namespace of the Role. By default its the
VirtualCluster's namespace",
                        required=True,
                        dest="namespace"
                        )

    parser.add_argument("-p", "--no-prompt",
                        help="Applies the patches without asking first",
                        dest="no_prompt",
                        default=False,
                        action="store_true"
                        )

    args = parser.parse_args()

    emrRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch"]
        }
    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch", "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch",
"deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["configmaps", "pods"],
            "verbs": ["deletecollection"]
        }
    ]

```

```
]

clientRoleRules = [
    {
        "apiGroups": [""],
        "resources": ["persistentvolumeclaims"],
        "verbs": ["list", "create", "delete", "patch"]
    }
]

patchRole("emr-containers", args.namespace, emrRoleRules, args.no_prompt)
patchRole("emr-containers-role-spark-driver", args.namespace, driverRoleRules,
args.no_prompt)
patchRole("emr-containers-role-spark-client", args.namespace, clientRoleRules,
args.no_prompt)
```

3. Ejecute el script de Python:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

4. Aparece una diferencia `kubectl` entre los permisos nuevos y los antiguos. Pulse y para parchear el rol.

5. Compruebe los tres roles con permisos adicionales de la siguiente manera:

```
kubectl describe role -n ${NAMESPACE}
```

6. Ejecute el script de Python:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

7. Después de ejecutar el comando, mostrará una diferencia de `kubectl` entre los permisos nuevos y los antiguos. Pulse y para parchear el rol.

8. Verifique los tres roles con permisos adicionales:

```
kubectl describe role -n ${NAMESPACE}
```

9. Vuelva a enviar el trabajo.

Parche manual

Si el permiso que requiere su aplicación se aplica a algo distinto a las reglas de PVC, puede agregar manualmente permisos de Kubernetes para su clúster virtual de Amazon EMR según sea necesario.

Note

El rol `emr-containers` es un rol principal. Esto significa que debe proporcionar todos los permisos necesarios antes de poder cambiar los roles subyacentes de controlador o cliente.

1. Descargue los permisos actuales en archivos yaml al ejecutar los siguientes comandos:

```
kubectl get role -n ${NAMESPACE} emr-containers -o yaml >> emr-containers-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-driver -o yaml >> driver-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-client -o yaml >> client-role-patch.yaml
```

2. En función del permiso que requiera su aplicación, edite cada archivo y agregue reglas adicionales, como las siguientes:

- `emr-containers-role-patch.yaml`

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

- `driver-role-patch.yaml`

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
```

```
- list
- create
- delete
- patch
- deletecollection
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
  - delete
  - watch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  verbs:
  - deletecollection
```

- client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

3. Elimine los siguientes atributos con sus valores. Esto es necesario para aplicar la actualización.

- creationTimestamp
- resourceVersion
- uid

4. Por último, ejecute el parche:

```
kubectl apply -f emr-containers-role-patch.yaml
kubectl apply -f driver-role-patch.yaml
kubectl apply -f client-role-patch.yaml
```

Solución de problemas en el escalado automático vertical de Amazon EMR en EKS

Consulte las siguientes secciones si tiene problemas al configurar el operador de escalado automático vertical de Amazon EMR en EKS en un clúster de Amazon EKS con Operator Lifecycle Manager. Para obtener más información, incluidos los pasos para completar la instalación, consulte [Uso del escalado automático vertical con trabajos de Spark de Amazon EMR](#).

Error 403 Prohibido

Si ha seguido los pasos indicados en [Instalar Operator Lifecycle Manager \(OLM\) en su clúster de Amazon EKS](#), ha ejecutado el comando `olm status` y le ha aparecido un error 403 Forbidden como el que se muestra a continuación, es posible que no haya obtenido los tokens de autenticación del operador en el repositorio de Amazon ECR.

Para resolver este problema, repita el paso de [Instalar el operador de escalado automático vertical de Amazon EMR en EKS](#) para obtener los tokens. A continuación, intente llevar a cabo la instalación de nuevo.

```
Error: FATA[0002] Failed to run bundle: pull bundle image: error pulling image IMAGE.
error resolving name : unexpected status code [manifests latest]: 403 Forbidden
```

No se encontró el espacio de nombres de Kubernetes

Al [configurar el operador de escalado automático vertical Amazon EMR en EKS](#) en un clúster de Amazon EKS, es posible que se produzca un error `namespaces not found` como el que se muestra aquí:

```
FATA[0020] Failed to run bundle: create catalog: error creating catalog source:
namespaces "NAME" not found.
```


Si el espacio de nombres que especifique no existe, OLM no instalará el operador de escalado automático vertical. Para resolver este problema, utilice el siguiente comando para crear el espacio de nombres. A continuación, intente llevar a cabo la instalación de nuevo.

```
kubectl create namespace NAME
```

Error al guardar las credenciales de Docker

Para [configurar el escalado automático vertical](#), debe autenticar y recuperar las imágenes de Docker verticales relacionadas con el escalado automático de Amazon EMR en EKS. Al hacer esto, es posible que aparezca un error como el siguiente si Docker no se está ejecutando:

```
aws ecr get-login-password \  
  --region $REGION | docker login \  
  --username AWS \  
  --password-stdin $ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com  
  
Error saving credentials: error storing credentials - err: exit status 1  
out: 'Post "http://ipc/registry/credstore-updated": dial unix backend.sock: connect: no  
such file or directory'
```

Para resolver este problema, confirme que Docker se está ejecutando o abra Docker Desktop. A continuación, intente guardar sus credenciales de nuevo.

Solución de problemas del operador de Spark de Amazon EMR en EKS

Consulte las siguientes secciones si tiene problemas con el operador de Spark de Amazon EMR en EKS. Para obtener más información, incluidos los pasos para completar la instalación, consulte [Ejecución de trabajos de Spark con el operador de Spark](#).

Error en la instalación del gráfico de Helm

Si siguió los pasos en [Instalar el operador de Spark](#) y le apareció un error INSTALLATION FAILED como el siguiente cuando intentó instalar o verificar el gráfico de Helm, es posible que no haya obtenido los tokens de autenticación en el repositorio de Amazon ECR para el operador.

Para resolver este problema, repita el paso de [Instalar el operador de Spark](#) para autenticar el cliente de Helm en el registro de Amazon ECR. A continuación, vuelva a intentar el paso de la instalación.

Error: INSTALLATION FAILED: Kubernetes cluster unreachable: the server has asked for the client to provide credentials

UnsupportedFileSystemException: No FileSystem para el esquema «s3»

Es posible que encuentre la siguiente excepción en el hilo “principal”:

```
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
```

Si esto ocurre, agregue las siguientes excepciones a la especificación SparkApplication:

```
hadoopConf:
  # EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
  mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
"2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
  # Required for EMR Runtime
  spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
  spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
  spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
  spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

Puntos de conexión de servicio y cuotas de Amazon EMR en EKS

A continuación se indican los puntos de conexión y las Service Quotas de Amazon EMR en EKS. Para conectarse mediante programación a un AWS servicio, se utiliza un punto final. Además de los AWS puntos finales estándar, algunos AWS servicios ofrecen puntos finales FIPS en determinadas regiones. Para obtener más información, consulte [puntos de conexión de servicio de AWS](#). Las Service Quotas, que también se denominan límites, establecen el número máximo de recursos u operaciones de servicio que puede haber en una cuenta de AWS . Para obtener más información, consulte [AWS Service Quotas](#).

Puntos de conexión de servicio

Región de AWS nombre	Código	Punto de conexión	Protocolo
Este de EE. UU. (Norte de Virginia)	us-east-1	emr-containers.us-east-1.amazonaws.com	HTTPS
Este de EE. UU. (Ohio)	us-east-2	emr-containers.us-east-2.amazonaws.com	HTTPS
Oeste de EE. UU. (Norte de California)	us-west-1	emr-containers.us-west-1.amazonaws.com	HTTPS
Oeste de EE. UU. (Oregón)	us-west-2	emr-containers.us-west-2.amazonaws.com	HTTPS
Asia-Pacífico (Tokio)	ap-northeast-1	emr-containers.ap-northeast-1.amazonaws.com	HTTPS
Asia-Pacífico (Seúl)	ap-northeast-2	emr-containers.ap-northeast-2.amazonaws.com	HTTPS
Asia-Pacífico (Osaka)	ap-northeast-3	emr-containers.ap-northeast-3.amazonaws.com	HTTPS

Región de AWS nombre	Código	Punto de conexión	Protocolo
Asia-Pacífico (Bombay)	ap-south-1	emr-containers.ap-south-1.amazonaws.com	HTTPS
Asia-Pacífico (Hyderabad)	ap-south-2	emr-containers.ap-south-2.amazonaws.com	HTTPS
Asia-Pacífico (Singapur)	ap-southeast-1	emr-containers.ap-southeast-1.amazonaws.com	HTTPS
Asia-Pacífico (Sídney)	ap-southeast-2	emr-containers.ap-southeast-2.amazonaws.com	HTTPS
Asia-Pacífico (Yakarta)	ap-southeast-3	emr-containers.ap-southeast-3.amazonaws.com	HTTPS
Asia-Pacífico (Hong Kong)	ap-east-1	emr-containers.ap-east-1.amazonaws.com	HTTPS
África (Ciudad del Cabo)	af-south-1	emr-containers.af-south-1.amazonaws.com	HTTPS
Canadá (centro)	ca-central-1	emr-containers.ca-central-1.amazonaws.com	HTTPS
China (Ningxia)	cn-northwest-1	emr-containers.cn-northwest-1.amazonaws.com.cn	HTTPS
China (Pekín)	cn-north-1	emr-containers.cn-north-1.amazonaws.com.cn	HTTPS
Europa (Fráncfort)	eu-central-1	emr-containers.eu-central-1.amazonaws.com	HTTPS
Europa (Zúrich)	eu-central-2	emr-containers.eu-central-2.amazonaws.com	HTTPS

Región de AWS nombre	Código	Punto de conexión	Protocolo
Europa (Irlanda)	eu-west-1	emr-containers.eu-west-1.amazonaws.com	HTTPS
Europa (Londres)	eu-west-2	emr-containers.eu-west-2.amazonaws.com	HTTPS
Europa (París)	eu-west-3	emr-containers.eu-west-3.amazonaws.com	HTTPS
Europa (Estocolmo)	eu-north-1	emr-containers.eu-north-1.amazonaws.com	HTTPS
Europa (Milán)	eu-south-1	emr-containers.eu-south-1.amazonaws.com	HTTPS
Europa (España)	eu-south-2	emr-containers.eu-south-2.amazonaws.com	HTTPS
Israel (Tel Aviv)	il-central-1	emr-containers.il-central-1.amazonaws.com	HTTPS
América del Sur (São Paulo)	sa-east-1	emr-containers.sa-east-1.amazonaws.com	HTTPS
Medio Oriente (EAU)	me-central-1	emr-containers.me-central-1.amazonaws.com	HTTPS
Medio Oriente (Baréin)	me-south-1	emr-containers.me-south-1.amazonaws.com	HTTPS
AWS GovCloud (Este de EE. UU.)	us-gov-east-1	emr-containers.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (Estados Unidos-Oeste)	us-gov-west-1	emr-containers.us-gov-west-1.amazonaws.com	HTTPS

Service Quotas

Amazon EMR en EKS limita las siguientes solicitudes de API para cada AWS cuenta por región. Para obtener más información sobre cómo se aplica la limitación, consulte [API Request Throttling](#) en la Amazon API Reference. EC2 Puedes solicitar un aumento de las cuotas de limitación de las API para tu AWS cuenta siguiendo la siguiente guía.

Acción de la API	Capacidad máxima del bucket	Velocidad de reposición del bucket (por segundo)
CancelJobRun	25	1
CreateJobTemplate	25	1
CreateManagedEndpoint	25	1
CreateSecurityConfiguration	25	1
CreateVirtualCluster	25	1
DeleteJobTemplate	25	1
DeleteManagedEndpoint	25	1
DeleteVirtualCluster	25	1
DescribeJobRun	100	20
DescribeJobTemplate	25	1
DescribeManagedEndpoint	100	5
DescribeSecurityConfiguration	25	1
DescribeVirtualCluster	100	5
GetManagedEndpoint SessionCredentials	25	1
ListJobRuns	100	5

Acción de la API	Capacidad máxima del bucket	Velocidad de reposición del bucket (por segundo)
ListJobTemplates	25	1
ListManagedEndpoints	25	1
ListSecurityConfigurations	25	1
ListVirtualClusters	100	5
StartJobRun	25	1
Throttle quota for all EMR on EKS API requests	200	20

Cuando creas tu AWS cuenta, establecemos las cuotas predeterminadas (también denominadas límites) para tus AWS recursos por región. Si intenta superar la cuota de un recurso, se producirá un error en la solicitud. Si esto ocurre, puede reducir el uso de recursos o solicitar un aumento de cuota.

La consola Service Quotas es una ubicación central donde puede ver y administrar las cuotas de AWS los servicios y solicitar aumentos de cuota para muchos de los recursos que utiliza. Utilice la información sobre cuotas que se proporciona aquí para administrar su AWS infraestructura. Planifique la solicitud del aumento de las cuotas antes del momento en que lo necesite.

Consulte las cuotas y solicite aumentos de cuota

Puede ver sus cuotas de servicio actuales para cada región mediante la consola Service Quotas. Para obtener instrucciones detalladas, consulte [Visualización de las cuotas de servicio](#) en la Guía del usuario de Service Quotas.

Puede solicitar un aumento de cuota desde la consola AWS de administración o mediante la AWS CLI. Los pasos se detallan en [Solicitar un aumento de cuota](#).

Versiones de Amazon EMR en EKS

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 Amazon EMR en EKS se implemente y configure cuando ejecute el trabajo.

A partir de las versiones 5.32.0 y 6.2.0 de Amazon EMR, puede implementar Amazon EMR en EKS. Esta opción de implementación no está disponible para versiones de lanzamiento anteriores de Amazon EMR. Debe especificar una versión de lanzamiento compatible al enviar el trabajo.

Amazon EMR en EKS utiliza la siguiente forma de etiqueta de versión: `emr-x.x.x-latest` o `emr-x.x.x-yyyyymmdd` con una fecha de versión específica. Por ejemplo, `emr-7.8.0-latest` o `emr-7.8.0-20210129`. Al utilizar el sufijo `-latest`, se asegura de que su versión de Amazon EMR siempre incluya las actualizaciones de seguridad más recientes.

Note

Para obtener una comparación entre Amazon EMR en EKS y Amazon EMR en ejecución, EC2 consulte Amazon [EMR](#) en el sitio web. FAQs AWS

Temas

- [Amazon EMR en las versiones 7.9.0 de EKS](#)
- [Amazon EMR en las versiones 7.8.0 de EKS](#)
- [Amazon EMR en las versiones 7.7.0 de EKS](#)
- [Amazon EMR en las versiones 7.6.0 de EKS](#)
- [Amazon EMR en las versiones 7.5.0 de EKS](#)
- [Amazon EMR en las versiones 7.4.0 de EKS](#)
- [Versiones 7.3.0 de Amazon EMR en EKS](#)
- [Versiones 7.2.0 de Amazon EMR en EKS](#)
- [Versiones 7.1.0 de Amazon EMR en EKS](#)
- [Versiones 7.0.0 de Amazon EMR en EKS](#)
- [Versiones 6.15.0 de Amazon EMR en EKS](#)

- [Versiones de Amazon EMR en EKS 6.14.0](#)
- [Versiones de Amazon EMR en EKS 6.13.0](#)
- [Versiones de Amazon EMR en EKS 6.12.0](#)
- [Versiones de Amazon EMR en EKS 6.11.0](#)
- [Versiones de Amazon EMR en EKS 6.10.0](#)
- [Versiones de Amazon EMR en EKS 6.9.0](#)
- [Versiones de Amazon EMR en EKS 6.8.0](#)
- [Versiones de Amazon EMR en EKS 6.7.0](#)
- [Versiones de Amazon EMR en EKS 6.6.0](#)
- [Versiones de Amazon EMR en EKS 6.5.0](#)
- [Versiones de Amazon EMR en EKS 6.4.0](#)
- [Versiones de Amazon EMR en EKS 6.3.0](#)
- [Versiones de Amazon EMR en EKS 6.2.0](#)
- [Versiones de Amazon EMR en EKS 5.36.0](#)
- [Versiones de Amazon EMR en EKS 5.35.0](#)
- [Versiones de Amazon EMR en EKS 5.34.0](#)
- [Versiones de Amazon EMR en EKS 5.33.0](#)
- [Versiones de Amazon EMR en EKS 5.32.0](#)

Amazon EMR en las versiones 7.9.0 de EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.9.0 de Amazon EMR en general, consulte [Amazon EMR 7.9.0 en la Guía de versiones de Amazon EMR](#).

Amazon EMR en las versiones 7.9 de EKS

Las siguientes versiones 7.9.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica del EMR-7.9.0-xxxx para ver más detalles, como la etiqueta de imagen del contenedor correspondiente.

Flink releases

Las siguientes versiones de Amazon EMR 7.9.0 están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Flink.

- [emr-7.9.0-Flink-última](#)
- [emr-7.9.0-flink-20250425](#)

Spark releases

Las siguientes versiones de Amazon EMR 7.9.0 están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Spark.

- [emr-7.9.0-latest](#)
- [emr-7.9.0-20250425](#)
- `emr-7.9.0-spark-rapids-latest`
- `emr-7.9.0-spark-rapids-20250425`
- `emr-7.9.0-java11-latest`
- `emr-7.9.0-java11-20250425`
- `emr-7.9.0-java8-latest`
- `emr-7.9.0-java8-20250425`
- `emr-7.9.0-spark-rapids-java8-latest`
- `emr-7.9.0-spark-rapids-java8-20250425`
- `notebook-spark/emr-7.9.0-latest`
- `notebook-spark/emr-7.9.0-20250425`
- `notebook-spark/emr-7.9.0-spark-rapids-latest`
- `notebook-spark/emr-7.9.0-spark-rapids-20250425`
- `notebook-spark/emr-7.9.0-java11-latest`
- `notebook-spark/emr-7.9.0-java11-20250425`
- `notebook-spark/emr-7.9.0-java8-latest`
- `notebook-spark/emr-7.9.0-java8-20250425`
- `notebook-spark/emr-7.9.0-spark-rapids-java8-latest`

- notebook-spark/emr-7.9.0-spark-rapids-java8-20250425
- notebook-python/emr-7.9.0-latest
- notebook-python/emr-7.9.0-20250425
- notebook-python/emr-7.9.0-spark-rapids-latest
- notebook-python/emr-7.9.0-spark-rapids-20250425
- notebook-python/emr-7.9.0-java11-latest
- notebook-python/emr-7.9.0-java11-20250425
- notebook-python/emr-7.9.0-java8-latest
- notebook-python/emr-7.9.0-java8-20250425
- notebook-python/emr-7.9.0-spark-rapids-java8-latest
- notebook-python/emr-7.9.0-spark-rapids-java8-20250425
- livy/emr-7.9.0-latest
- livy/emr-7.9.0-20250425
- livy/emr-7.9.0-java11-latest
- livy/emr-7.9.0-java11-20250425
- livy/emr-7.9.0-java8-latest
- livy/emr-7.9.0-java8-20250425

Notas de la versión

Notas de publicación de Amazon EMR en EKS 7.9.0

- Aplicaciones compatibles: AWS SDK para Java 2.31.16 and 1.12.782, Apache Spark 3.5.5, Apache Hudi 0.15.0-amzn-6, Apache Iceberg 1.7.1-amzn-2, Delta 3.3.0-amzn-1, Apache Spark RAPIDS 25.02.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-3, Flink Operator 1.10.0-amzn-3
- Componentes compatibles -emr-ddb,emr-goodies,emr-s3-select,emrfs,hadoop-client, hudihudi-spark,iceberg. spark-kubernetes
- Clasificaciones de configuración compatibles

Para usar con [StartJobRuny](#) [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j2	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Cambios

La versión 7.9.0 de Amazon EMR en EKS incluye los siguientes cambios:

- No hay cambios en la versión.

emr-7.9.0-latest

Notas de la versión: actualmente, `emr-7.9.0-latest` apunta a `emr-7.9.0-20250425`.

Regiones: `emr-7.9.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.9.0:latest`

emr-7.9.0-20250425

Notas de lanzamiento: se lanzó en febrero de 2025. `emr-7.9.0-20250425` Esta es la versión inicial de Amazon EMR 7.9.0 (Spark).

Regiones: `emr-emr-7.9.0-20250425` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.9.0-20250425`

emr-7.9.0-Flink-última

Notas de la versión: actualmente apunta a `emr-7.9.0-flink-latest` `emr-7.9.0-flink-20250425`

Regiones: `emr-7.9.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.9.0-flink:latest`

emr-7.9.0-flink-20250425

Notas de lanzamiento: se lanzó en febrero de 2025. `7.9.0-flink-20250425` Esta es la versión inicial de Amazon EMR 7.9.0 (Flink).

Regiones: `emr-7.9.0-flink-20250425` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.9.0-flink:20250425`

Amazon EMR en las versiones 7.8.0 de EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.8.0 de Amazon EMR en general, consulte [Amazon EMR 7.8.0 en la Guía de versiones de Amazon EMR](#).

Amazon EMR en las versiones 7.8 de EKS

Las siguientes versiones de Amazon EMR 7.8.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica del `EMR-7.8.0-xxxx` para ver más detalles, como la etiqueta de imagen del contenedor correspondiente.

Flink releases

Las siguientes versiones de Amazon EMR 7.8.0 están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Flink.

- [emr-7.8.0-flink-latest](#)
- [emr-7.8.0-flink-20250228](#)

Spark releases

Las siguientes versiones de Amazon EMR 7.8.0 están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Spark.

- [emr-7.8.0-último](#)
- [emr-7.8.0-20250228](#)
- `emr-7.8.0-spark-rapids-latest`
- `emr-7.8.0-spark-rapids-20250228`
- `emr-7.8.0-java11-latest`
- `emr-7.8.0-java11-20250228`
- `emr-7.8.0-java8-latest`
- `emr-7.8.0-java8-20250228`
- `emr-7.8.0-spark-rapids-java8-latest`
- `emr-7.8.0-spark-rapids-java8-20250228`
- `notebook-spark/emr-7.8.0-latest`
- `notebook-spark/emr-7.8.0-20250228`
- `notebook-spark/emr-7.8.0-spark-rapids-latest`
- `notebook-spark/emr-7.8.0-spark-rapids-20250228`
- `notebook-spark/emr-7.8.0-java11-latest`
- `notebook-spark/emr-7.8.0-java11-20250228`
- `notebook-spark/emr-7.8.0-java8-latest`
- `notebook-spark/emr-7.8.0-java8-20250228`
- `notebook-spark/emr-7.8.0-spark-rapids-java8-latest`
- `notebook-spark/emr-7.8.0-spark-rapids-java8-20250228`
- `notebook-python/emr-7.8.0-latest`
- `notebook-python/emr-7.8.0-20250228`
- `notebook-python/emr-7.8.0-spark-rapids-latest`
- `notebook-python/emr-7.8.0-spark-rapids-20250228`
- `notebook-python/emr-7.8.0-java11-latest`
- `notebook-python/emr-7.8.0-java11-20250228`
- `notebook-python/emr-7.8.0-java8-latest`
- `notebook-python/emr-7.8.0-java8-20250228`
- `notebook-python/emr-7.8.0-spark-rapids-java8-latest`

- notebook-python/emr-7.8.0-spark-rapids-java8-20250228
- livy/emr-7.8.0-latest
- livy/emr-7.8.0-20250228
- livy/emr-7.8.0-java11-latest
- livy/emr-7.8.0-java11-20250228
- livy/emr-7.8.0-java8-latest
- livy/emr-7.8.0-java8-20250228

Notas de la versión

Notas de publicación de Amazon EMR en EKS 7.8.0

- Aplicaciones compatibles: AWS SDK para Java 2.29.52 and 1.12.780, Apache Spark 3.5.4, Apache Hudi 0.15.0-amzn-5, Apache Iceberg 1.7.1-amzn-1, Delta 3.3.0-amzn-0, Apache Spark RAPIDS 24.12.0-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-2, Flink Operator 1.10.0-amzn-2
- Componentes compatibles -emr-ddb,emr-goodies,emr-s3-select,emrfs,hadoop-client, hudi-hudi-spark,iceberg. spark-kubernetes
- Clasificaciones de configuración compatibles

Para usar con [StartJobRuny](#) [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.

Clasificaciones	Descripciones
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j2	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Cambios

La versión 7.8.0 de Amazon EMR en EKS incluye los siguientes cambios:

- Características del FGAC nativo, que incluyen:
 - Soporte Iceberg para ejecutar trabajos que realizan acciones en tablas que no son de formación de lagos en un clúster virtual de control de acceso (FGAC) detallado. (Existe una alternativa a la IAM).
 - Soporte para tablas S3

- Spark connect

emr-7.8.0-último

Notas de la versión: actualmente, `emr-7.8.0-latest` apunta a `emr-7.8.0-20250228`.

Regiones: `emr-7.8.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.8.0:latest`

emr-7.8.0-20250228

Notas de lanzamiento: se lanzó en febrero de 2025. `emr-7.8.0-20250228` Esta es la versión inicial de Amazon EMR 7.8.0 (Spark).

Regiones: `emr-emr-7.8.0-20250228` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.8.0-20250228`

emr-7.8.0-flink-latest

Notas de la versión: actualmente apunta a `emr-7.8.0-flink-latest` `emr-7.8.0-flink-20250228`

Regiones: `emr-7.8.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.8.0-flink:latest`

emr-7.8.0-flink-20250228

Notas de lanzamiento: se lanzó en febrero de 2025. `7.8.0-flink-20250228` Esta es la versión inicial de Amazon EMR 7.8.0 (Flink).

Regiones: `emr-7.8.0-flink-20250228` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.8.0-flink:20250228`

Amazon EMR en las versiones 7.7.0 de EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.7.0 de Amazon EMR en general, consulte [Amazon EMR 7.7.0 en la Guía de versiones de Amazon EMR](#).

Amazon EMR en las versiones 7.7 de EKS

Las siguientes versiones de Amazon EMR 7.7.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica del `EMR-7.7.0-xxxx` para ver más detalles, como la etiqueta de imagen del contenedor correspondiente.

Flink releases

Las siguientes versiones de Amazon EMR 7.7.0 están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Flink.

- [emr-7.7.0-Flink-latest](#)
- [emr-7.7.0-flink-20250131](#)

Spark releases

Las siguientes versiones de Amazon EMR 7.7.0 están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Spark.

- [emr-7.7.0-latest](#)
- [emr-7.7.0-20250131](#)
- `emr-7.7.0-spark-rapids-latest`
- `emr-7.7.0-spark-rapids-20250131`
- `emr-7.7.0-java11-latest`
- `emr-7.7.0-java11-20250131`

- `emr-7.7.0-java8-latest`
- `emr-7.7.0-java8-20250131`
- `emr-7.7.0-spark-rapids-java8-latest`
- `emr-7.7.0-spark-rapids-java8-20250131`
- `notebook-spark/emr-7.7.0-latest`
- `notebook-spark/emr-7.7.0-20250131`
- `notebook-spark/emr-7.7.0-spark-rapids-latest`
- `notebook-spark/emr-7.7.0-spark-rapids-20250131`
- `notebook-spark/emr-7.7.0-java11-latest`
- `notebook-spark/emr-7.7.0-java11-20250131`
- `notebook-spark/emr-7.7.0-java8-latest`
- `notebook-spark/emr-7.7.0-java8-20250131`
- `notebook-spark/emr-7.7.0-spark-rapids-java8-latest`
- `notebook-spark/emr-7.7.0-spark-rapids-java8-20250131`
- `notebook-python/emr-7.7.0-latest`
- `notebook-python/emr-7.7.0-20250131`
- `notebook-python/emr-7.7.0-spark-rapids-latest`
- `notebook-python/emr-7.7.0-spark-rapids-20250131`
- `notebook-python/emr-7.7.0-java11-latest`
- `notebook-python/emr-7.7.0-java11-20250131`
- `notebook-python/emr-7.7.0-java8-latest`
- `notebook-python/emr-7.7.0-java8-20250131`
- `notebook-python/emr-7.7.0-spark-rapids-java8-latest`
- `notebook-python/emr-7.7.0-spark-rapids-java8-20250131`
- `livy/emr-7.7.0-latest`
- `livy/emr-7.7.0-20250131`
- `livy/emr-7.7.0-java11-latest`
- `livy/emr-7.7.0-java11-20250131`
- `livy/emr-7.7.0-java8-latest`
- `livy/emr-7.7.0-java8-20250131`

Notas de la versión

Notas de publicación de Amazon EMR en EKS 7.7.0

- Aplicaciones compatibles: AWS SDK para Java 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- Componentes compatibles -emr-ddb,emr-goodies,emr-s3-select,emrfs,hadoop-client, hudi-hudi-spark,iceberg. spark-kubernetes
- Clasificaciones de configuración compatibles

Para usar con [StartJobRuny](#) [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop core-site.xml .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark metrics.properties .
spark-defaults	Cambia los valores en el archivo de Spark spark-defaults.conf .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark hive-site.xml .
spark-log4j2	Cambia los valores en el archivo de Spark log4j2.properties .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Cambios

La versión 7.7.0 de Amazon EMR en EKS incluye los siguientes cambios:

- La versión de Iceberg que se utilizaba a partir de EMR 7.7.0 ya no es compatible con Java 8. Además, Iceberg está excluido de las siguientes imágenes de Java 8: `y. emr-7.7.0-java8-latest` `emr-7.7.0-spark-rapids-java8-latest`

emr-7.7.0-latest

Notas de la versión: actualmente, `emr-7.7.0-latest` apunta a `emr-7.7.0-20250131`.

Regiones: `emr-7.7.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.7.0:latest`

emr-7.7.0-20250131

Notas de lanzamiento: se lanzó en febrero de 2025. `emr-7.7.0-20250131` Esta es la versión inicial de Amazon EMR 7.7.0 (Spark).

Regiones: `emr-emr-7.7.0-20250131` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.7.0-20250131`

emr-7.7.0-Flink-latest

Notas de la versión: actualmente apunta a `emr-7.7.0-flink-latest` `emr-7.7.0-flink-20250131`

Regiones: `emr-7.7.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.7.0-flink:latest`

emr-7.7.0-flink-20250131

Notas de lanzamiento: se lanzó en febrero de 2025. `7.7.0-flink-20250131` Esta es la versión inicial de Amazon EMR 7.7.0 (Flink).

Regiones: `emr-7.7.0-flink-20250131` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.7.0-flink:20250131`

Amazon EMR en las versiones 7.6.0 de EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.6.0 de Amazon EMR en general, consulte [Amazon EMR 7.6.0 en la Guía de versiones de Amazon EMR](#).

Amazon EMR en las versiones 7.6 de EKS

Las siguientes versiones 7.6.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica del EMR-7.6.0-xxxx para ver más detalles, como la etiqueta de imagen del contenedor correspondiente.

Flink releases

Las siguientes versiones 7.6.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Flink.

- [emr-7.6.0-flink-latest](#)
- [emr-7.6.0-flink-20241213](#)

Spark releases

Las siguientes versiones 7.6.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones Spark.

- [emr-7.6.0-latest](#)
- [emr-7.6.0-20241213](#)
- `emr-7.6.0-spark-rapids-latest`
- `emr-7.6.0-spark-rapids-20241213`
- `emr-7.6.0-java11-latest`
- `emr-7.6.0-java11-20241213`
- `emr-7.6.0-java8-latest`
- `emr-7.6.0-java8-20241213`
- `emr-7.6.0-spark-rapids-java8-latest`
- `emr-7.6.0-spark-rapids-java8-20241213`
- `notebook-spark/emr-7.6.0-latest`
- `notebook-spark/emr-7.6.0-20241213`
- `notebook-spark/emr-7.6.0-spark-rapids-latest`
- `notebook-spark/emr-7.6.0-spark-rapids-20241213`
- `notebook-spark/emr-7.6.0-java11-latest`
- `notebook-spark/emr-7.6.0-java11-20241213`

- notebook-spark/emr-7.6.0-java8-latest
- notebook-spark/emr-7.6.0-java8-20241213
- notebook-spark/emr-7.6.0-spark-rapids-java8-latest
- notebook-spark/emr-7.6.0-spark-rapids-java8-20241213
- notebook-python/emr-7.6.0-latest
- notebook-python/emr-7.6.0-20241213
- notebook-python/emr-7.6.0-spark-rapids-latest
- notebook-python/emr-7.6.0-spark-rapids-20241213
- notebook-python/emr-7.6.0-java11-latest
- notebook-python/emr-7.6.0-java11-20241213
- notebook-python/emr-7.6.0-java8-latest
- notebook-python/emr-7.6.0-java8-20241213
- notebook-python/emr-7.6.0-spark-rapids-java8-latest
- notebook-python/emr-7.6.0-spark-rapids-java8-20241213
- livy/emr-7.6.0-latest
- livy/emr-7.6.0-20241213
- livy/emr-7.6.0-java11-latest
- livy/emr-7.6.0-java11-20241213
- livy/emr-7.6.0-java8-latest
- livy/emr-7.6.0-java8-20241213

Notas de la versión

Notas de publicación de Amazon EMR en EKS 7.6.0

- Aplicaciones compatibles: AWS SDK para Java 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

Para usar con [StartJobRuny](#): [CreateManagedEndpoint](#) APIs

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j2	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

La versión 7.6.0 de Amazon EMR en EKS incluye las siguientes funciones:

- Soporte de configuración de monitoreo para Apache Spark Operator: la configuración de monitoreo le permite configurar fácilmente el archivado de registros de su aplicación Spark y los registros del operador en Amazon S3 o Amazon CloudWatch. Puede elegir uno o ambos. De este modo, se añade un sidecar de agente de registro a los módulos de operador, controlador y ejecutor de Spark y, posteriormente, se reenvían los registros de estos componentes a los sumideros configurados. Para obtener más información, consulta [Cómo usar la configuración de monitoreo para monitorear el operador de Spark Kubernetes y las tareas de Spark](#).

Cambios

La versión 7.6.0 de Amazon EMR en EKS incluye los siguientes cambios:

- No hay cambios en la versión.

emr-7.6.0-latest

Notas de la versión: actualmente, `emr-7.6.0-latest` apunta a `emr-7.6.0-20241213`.

Regiones: `emr-7.6.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.6.0:latest`

emr-7.6.0-20241213

Notas de lanzamiento: se lanzó en enero de 2024. `7.6.0-20241213` Esta es la versión inicial de Amazon EMR 7.6.0 (Spark).

Regiones: `emr-7.6.0-20241213` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.6.0:20241213`

`emr-7.6.0-flink-latest`

Notas de la versión: actualmente apunta a `emr-7.6.0-flink-latest` `emr-7.6.0-flink-20241213`

Regiones: `emr-7.6.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.6.0-flink:latest`

`emr-7.6.0-flink-20241213`

Notas de lanzamiento: se lanzó en enero de 2024. `7.6.0-flink-20241213` Esta es la versión inicial de Amazon EMR 7.6.0 (Flink).

Regiones: `emr-7.6.0-flink-20241213` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.6.0-flink:20241213`

Amazon EMR en las versiones 7.5.0 de EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.5.0 de Amazon EMR en general, consulte [Amazon EMR 7.5.0 en la Guía de versiones de Amazon EMR](#).

Amazon EMR en las versiones 7.5 de EKS

Las siguientes versiones de Amazon EMR 7.5.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica del `EMR-7.5.0-xxxx` para ver más detalles, como la etiqueta de imagen del contenedor correspondiente.

Notas de la versión

Notas de publicación de Amazon EMR en EKS 7.5.0

- Aplicaciones compatibles: AWS SDK para Java 2.28.8 and 1.12.772, Apache Spark 3.5.2-amzn-1, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-1, Flink Operator 1.9.0-amzn-0
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.

Amazon EMR en las versiones 7.4.0 de EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.4.0 de Amazon en general, consulte [Amazon EMR 7.4.0 en la Guía de versiones de Amazon EMR](#).

Amazon EMR en las versiones 7.4 de EKS

Las siguientes versiones de Amazon EMR 7.4.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica del EMR-7.4.0-xxxx para ver más detalles, como la etiqueta de imagen del contenedor correspondiente.

Notas de la versión

Notas de publicación de Amazon EMR en EKS 7.4.0

- Aplicaciones compatibles: AWS SDK para Java 2.25.70 and 1.12.772, Apache Spark 3.5.2-amzn-0, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-0, Flink Operator 1.9.0-amzn-1
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.

Versiones 7.3.0 de Amazon EMR en EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.3.0 de Amazon EMR en general, consulte [Amazon EMR 7.3.0 en la Guía de versiones de Amazon EMR](#).

Versiones 7.3 de Amazon EMR en EKS

Las siguientes versiones 7.3.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-7.3.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

Flink releases

Las siguientes versiones 7.3.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Flink.

- [emr-7.3.0-flink-latest](#)
- [emr-7.3.0-flink-29240920](#)

Spark releases

Las siguientes versiones 7.3.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Spark.

- [emr-7.3.0-latest](#)
- [emr-7.3.0-29240920](#)
- `emr-7.3.0-spark-rapids-latest`
- `emr-7.3.0-spark-rapids-29240920`
- `emr-7.3.0-java11-latest`
- `emr-7.3.0-java11-29240920`
- `emr-7.3.0-java8-latest`
- `emr-7.3.0-java8-29240920`
- `emr-7.3.0-spark-rapids-java8-latest`
- `emr-7.3.0-spark-rapids-java8-29240920`

- notebook-spark/emr-7.3.0-latest
- notebook-spark/emr-7.3.0-29240920
- notebook-spark/emr-7.3.0-spark-rapids-latest
- notebook-spark/emr-7.3.0-spark-rapids-29240920
- notebook-spark/emr-7.3.0-java11-latest
- notebook-spark/emr-7.3.0-java11-29240920
- notebook-spark/emr-7.3.0-java8-latest
- notebook-spark/emr-7.3.0-java8-29240920
- notebook-spark/emr-7.3.0-spark-rapids-java8-latest
- notebook-spark/emr-7.3.0-spark-rapids-java8-29240920
- notebook-python/emr-7.3.0-latest
- notebook-python/emr-7.3.0-29240920
- notebook-python/emr-7.3.0-spark-rapids-latest
- notebook-python/emr-7.3.0-spark-rapids-29240920
- notebook-python/emr-7.3.0-java11-latest
- notebook-python/emr-7.3.0-java11-29240920
- notebook-python/emr-7.3.0-java8-latest
- notebook-python/emr-7.3.0-java8-29240920
- notebook-python/emr-7.3.0-spark-rapids-java8-latest
- notebook-python/emr-7.3.0-spark-rapids-java8-29240920
- livy/emr-7.3.0-latest
- livy/emr-7.3.0-29240920
- livy/emr-7.3.0-java11-latest
- livy/emr-7.3.0-java11-29240920
- livy/emr-7.3.0-java8-latest
- livy/emr-7.3.0-java8-29240920

Notas de la versión

Notas de la versión 7.3.0 de Amazon EMR en EKS

- Aplicaciones compatibles: AWS SDK para Java 2.25.70 and 1.12.747, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.15.0-amzn-0, Apache Iceberg 1.5.2-amzn-0, Delta 3.2.0-amzn-0, Apache Spark RAPIDS 24.06.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-2, Flink Operator 1.9.0-amzn-0
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

Para usar con y: [StartJobRun CreateManagedEndpoint](#) APIs

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j2	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 7.3.0 de Amazon EMR en EKS.

- Actualizaciones de aplicaciones: Amazon EMR en EKS ahora incluye [Flink Operator](#) 1.9.0. Además de otras funciones, el Flink Kubernetes ahora le permite establecer las cuotas de CPU y memoria para el escalador automático.
- Soporte de Apache Iceberg para Apache Flink: Apache Iceberg es un formato de código abierto de alto rendimiento para tablas de análisis de gran tamaño. A partir de Amazon EMR 7.3.0, puede usar tablas de Apache Iceberg al ejecutar Apache Flink en Amazon EMR en EKS. Para obtener más información, consulte [Uso de Apache Iceberg con Amazon EMR en EKS](#) del Amazon EMR en EKS.
- Compatibilidad de Delta Lake para Apache Flink: Delta Lake es un marco de capas de almacenamiento para arquitecturas Lakehouse que se suelen basar en Amazon S3. Con Amazon EMR 7.3.0 y versiones posteriores, puede usar tablas Delta al ejecutar Apache Flink en Amazon EMR en EKS. Para obtener más información, consulte [Uso de Delta Lake con Amazon EMR en EKS](#).

Cambios

Los siguientes cambios se incluyen en la versión 7.3.0 de Amazon EMR en EKS.

- Con Amazon EMR en EKS 7.3.0 y versiones posteriores, Apache Flink ahora usa el tiempo de ejecución Java 17 de forma predeterminada.

emr-7.3.0-latest

Notas de la versión: actualmente, `emr-7.3.0-latest` apunta a `emr-7.3.0-29240920`.

Regiones: `emr-7.3.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.3.0:latest`

emr-7.3.0-29240920

Notas de la versión: `7.3.0-29240920` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.3.0 (Spark).

Regiones: `emr-7.3.0-29240920` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.3.0:29240920`

emr-7.3.0-flink-latest

Notas de la versión: actualmente, `emr-7.3.0-flink-latest` apunta a `emr-7.3.0-flink-29240920`.

Regiones: `emr-7.3.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.3.0-flink:latest`

emr-7.3.0-flink-29240920

Notas de la versión: `7.3.0-flink-29240920` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.3.0 (Flink).

Regiones: `emr-7.3.0-flink-29240920` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.3.0-flink:29240920`

Versiones 7.2.0 de Amazon EMR en EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.2.0 de Amazon EMR en general, consulte [Amazon EMR 7.2.0 en la Guía de versiones de Amazon EMR](#).

Versiones 7.2 de Amazon EMR en EKS

Las siguientes versiones 7.2.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-7.2.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

Flink releases

Las siguientes versiones 7.2.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Flink.

- [emr-7.2.0-flink-latest](#)
- [emr-7.2.0-flink-20240610](#)

Spark releases

Las siguientes versiones 7.2.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Spark.

- [emr-7.2.0-latest](#)

- [emr-7.2.0-20240610](#)
- `emr-7.2.0-spark-rapids-latest`
- `emr-7.2.0-spark-rapids-20240610`
- `emr-7.2.0-java11-latest`
- `emr-7.2.0-java11-20240610`
- `emr-7.2.0-java8-latest`
- `emr-7.2.0-java8-20240610`
- `emr-7.2.0-spark-rapids-java8-latest`
- `emr-7.2.0-spark-rapids-java8-20240610`
- `notebook-spark/emr-7.2.0-latest`
- `notebook-spark/emr-7.2.0-20240610`
- `notebook-spark/emr-7.2.0-spark-rapids-latest`
- `notebook-spark/emr-7.2.0-spark-rapids-20240610`
- `notebook-spark/emr-7.2.0-java11-latest`
- `notebook-spark/emr-7.2.0-java11-20240610`
- `notebook-spark/emr-7.2.0-java8-latest`
- `notebook-spark/emr-7.2.0-java8-20240610`
- `notebook-spark/emr-7.2.0-spark-rapids-java8-latest`
- `notebook-spark/emr-7.2.0-spark-rapids-java8-20240610`
- `notebook-python/emr-7.2.0-latest`
- `notebook-python/emr-7.2.0-20240610`
- `notebook-python/emr-7.2.0-spark-rapids-latest`
- `notebook-python/emr-7.2.0-spark-rapids-20240610`
- `notebook-python/emr-7.2.0-java11-latest`
- `notebook-python/emr-7.2.0-java11-20240610`
- `notebook-python/emr-7.2.0-java8-latest`
- `notebook-python/emr-7.2.0-java8-20240610`
- `notebook-python/emr-7.2.0-spark-rapids-java8-latest`

- notebook-python/emr-7.2.0-spark-rapids-java8-20240610
- livy/emr-7.2.0-latest
- livy/emr-7.2.0-20240610
- livy/emr-7.2.0-java11-latest
- livy/emr-7.2.0-java11-20240610
- livy/emr-7.2.0-java8-latest
- livy/emr-7.2.0-java8-20240610

Notas de la versión

Notas de la versión 7.2.0 de Amazon EMR en EKS

- Aplicaciones compatibles: AWS SDK para Java 2.23.18 and 1.12.705, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.5.0-amzn-0, Delta 3.1.0, Apache Spark RAPIDS 24.02.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.8.0-amzn-1
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

Para usar con y: [StartJobRun CreateManagedEndpoint](#) APIs

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop core-site.xml .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark metrics.properties .
spark-defaults	Cambia los valores en el archivo de Spark spark-defaults.conf .
spark-env	Cambiar los valores en el entorno de Spark.

Clasificaciones	Descripciones
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j2	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 7.2.0 de Amazon EMR en EKS.

- Actualizaciones de la aplicación: las actualizaciones de la aplicación de Amazon EMR en EKS 7.2.0 incluyen Spark 3.5.1, Flink 1.18.1 y [Flink Operator](#) 1.8.0.
- [Actualizaciones del escalador automático para Flink](#): la versión 7.2.0 utiliza la configuración de código abierto `job.autoscaler.restart.time-tracking.enabled` para permitir el cálculo del tiempo de reescalado, por lo que ya no es necesario asignar manualmente valores empíricos

al tiempo de reinicio. Si ejecuta la versión 7.1.0 o una versión anterior, puede seguir utilizando el escalado automático de Amazon EMR.

- [Integración de Apache Hudi Apache Flink en Amazon EMR en EKS](#): esta versión añade una integración entre Apache Hudi y Apache Flink, de modo que puede utilizar el operador Flink Kubernetes para ejecutar trabajos de Hudi. Hudi le permite utilizar operaciones a nivel de registro que puede utilizar para simplificar la administración de datos y el desarrollo de la canalización de datos.
- [Integración de Amazon S3 Express One Zone con Amazon EMR en EKS](#): con la versión 7.2.0 y versiones posteriores, puede cargar datos en S3 Express One Zone con Amazon EMR en EKS. 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.
- [Compatibilidad con las configuraciones predeterminadas en el operador Spark](#): el operador Spark de Amazon EKS ahora admite las mismas configuraciones predeterminadas que el modelo de ejecución de trabajos de inicio en Amazon EMR en EKS para la versión 7.2.0 y versiones posteriores. Esto significa que funciones como Amazon S3 y EMRFS ya no requieren configuraciones manuales en el archivo yaml.

emr-7.2.0-latest

Notas de la versión: actualmente, `emr-7.2.0-latest` apunta a `emr-7.2.0-20240610`.

Regiones: `emr-7.2.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.2.0:latest`

emr-7.2.0-20240610

Notas de la versión: `7.2.0-20240610` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.2.0 (Spark).

Regiones: `emr-7.2.0-20240610` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.2.0:20240610`

`emr-7.2.0-flink-latest`

Notas de la versión: actualmente, `emr-7.2.0-flink-latest` apunta a `emr-7.2.0-flink-20240610`.

Regiones: `emr-7.2.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.2.0-flink:latest`

`emr-7.2.0-flink-20240610`

Notas de la versión: `7.2.0-flink-20240610` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.2.0 (Flink).

Regiones: `emr-7.2.0-flink-20240610` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.2.0-flink:20240610`

Versiones 7.1.0 de Amazon EMR en EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.1.0 de Amazon EMR en general, consulte [Amazon EMR 7.1.0 en la Guía de versiones de Amazon EMR](#).

Versiones 7.1 de Amazon EMR en EKS

Las siguientes versiones 7.1.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-7.1.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

Flink releases

Las siguientes versiones 7.1.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Flink.

- [emr-7.1.0-flink-latest](#)
- [emr-7.1.0-flink-20240321](#)

Spark releases

Las siguientes versiones 7.1.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Spark.

- [emr-7.1.0-latest](#)
- [emr-7.1.0-20240321](#)
- `emr-7.1.0-spark-rapids-latest`
- `emr-7.1.0-spark-rapids-20240321`
- `emr-7.1.0-java11-latest`
- `emr-7.1.0-java11-20240321`
- `emr-7.1.0-java8-latest`
- `emr-7.1.0-java8-20240321`
- `emr-7.1.0-spark-rapids-java8-latest`
- `emr-7.1.0-spark-rapids-java8-20240321`
- `notebook-spark/emr-7.1.0-latest`
- `notebook-spark/emr-7.1.0-20240321`
- `notebook-spark/emr-7.1.0-spark-rapids-latest`
- `notebook-spark/emr-7.1.0-spark-rapids-20240321`
- `notebook-spark/emr-7.1.0-java11-latest`
- `notebook-spark/emr-7.1.0-java11-20240321`
- `notebook-spark/emr-7.1.0-java8-latest`
- `notebook-spark/emr-7.1.0-java8-20240321`
- `notebook-spark/emr-7.1.0-spark-rapids-java8-latest`
- `notebook-spark/emr-7.1.0-spark-rapids-java8-20240321`
- `notebook-python/emr-7.1.0-latest`
- `notebook-python/emr-7.1.0-20240321`
- `notebook-python/emr-7.1.0-spark-rapids-latest`

- notebook-python/emr-7.1.0-spark-rapids-20240321
- notebook-python/emr-7.1.0-java11-latest
- notebook-python/emr-7.1.0-java11-20240321
- notebook-python/emr-7.1.0-java8-latest
- notebook-python/emr-7.1.0-java8-20240321
- notebook-python/emr-7.1.0-spark-rapids-java8-latest
- notebook-python/emr-7.1.0-spark-rapids-java8-20240321
- livy/emr-7.1.0-latest
- livy/emr-7.1.0-20240321
- livy/emr-7.1.0-java11-latest
- livy/emr-7.1.0-java11-20240321
- livy/emr-7.1.0-java8-latest
- livy/emr-7.1.0-java8-20240321

Notas de la versión

Notas de la versión 7.1.0 de Amazon EMR en EKS

- Aplicaciones compatibles: AWS SDK para Java 2.23.18 and 1.12.656, Apache Spark 3.5.0-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.4.3-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.6.1-amzn-1
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

Para usar con y: [StartJobRun CreateManagedEndpoint](#) APIs

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop core-site.xml .
emrfs-site	Cambiar la configuración de EMRFS.

Clasificaciones	Descripciones
<code>spark-metrics</code>	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
<code>spark-defaults</code>	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
<code>spark-log4j2</code>	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
<code>emr-job-submitter</code>	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
<code>jeg-config</code>	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
<code>jupyter-kernel-overrides</code>	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 7.1.0 de Amazon EMR en EKS.

- [Compatibilidad de Apache Livy para Amazon EMR en EKS](#): con Amazon EMR en EKS en las versiones 7.1.0 y posteriores, puede usar Apache Livy en un clúster de Amazon EKS para crear una interfaz REST de Apache Livy y enviar trabajos de Spark o fragmentos de código de Spark. De este modo, podrá recuperar los resultados de forma sincrónica y asíncrona y, al mismo tiempo, aprovechar las ventajas de Amazon EMR en EKS, como el tiempo de ejecución de Spark optimizado para Amazon EMR, los puntos de conexión Livy habilitados para SSL y una experiencia de configuración programática.

emr-7.1.0-latest

Notas de la versión: actualmente, `emr-7.1.0-latest` apunta a `emr-7.1.0-20240321`.

Regiones: `emr-7.1.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.1.0:latest`

emr-7.1.0-20240321

Notas de la versión: `7.1.0-20240321` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.1.0 (Spark).

Regiones: `emr-7.1.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.1.0:20240321`

emr-7.1.0-flink-latest

Notas de la versión: actualmente, `emr-7.1.0-flink-latest` apunta a `emr-7.1.0-flink-20240321`.

Regiones: `emr-7.1.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.1.0-flink:latest`

`emr-7.1.0-flink-20240321`

Notas de la versión: `7.1.0-flink-20240321` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.1.0 (Flink).

Regiones: `emr-7.1.0-flink-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.1.0-flink:20240321`

Versiones 7.0.0 de Amazon EMR en EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 7.0.0 de Amazon EMR en general, consulte [Amazon EMR 7.0.0 en la Guía de versiones de Amazon EMR](#).

Versiones 7.0 de Amazon EMR en EKS

Las siguientes versiones 7.0.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-7.0.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

Flink releases

Las siguientes versiones 7.0.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Flink.

- [emr-7.0.0-flink-latest](#)
- [emr-7.0.0-flink-2024321](#)
- [emr-7.0.0-flink-20231211](#)

Spark releases

Las siguientes versiones 7.0.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Spark.

- [emr-7.0.0-latest](#)
- [emr-7.0.0-20231211](#)
- `emr-7.0.0-spark-rapids-latest`
- `emr-7.0.0-spark-rapids-20231211`
- `emr-7.0.0-java11-latest`
- `emr-7.0.0-java11-20231211`
- `emr-7.0.0-java8-latest`
- `emr-7.0.0-java8-20231211`
- `emr-7.0.0-spark-rapids-java8-latest`
- `emr-7.0.0-spark-rapids-java8-20231211`
- `notebook-spark/emr-7.0.0-latest`
- `notebook-spark/emr-7.0.0-20231211`
- `notebook-spark/emr-7.0.0-spark-rapids-latest`
- `notebook-spark/emr-7.0.0-spark-rapids-20231211`
- `notebook-spark/emr-7.0.0-java11-latest`
- `notebook-spark/emr-7.0.0-java11-20231211`
- `notebook-spark/emr-7.0.0-java8-latest`
- `notebook-spark/emr-7.0.0-java8-20231211`
- `notebook-spark/emr-7.0.0-spark-rapids-java8-latest`
- `notebook-spark/emr-7.0.0-spark-rapids-java8-20231211`
- `notebook-python/emr-7.0.0-latest`
- `notebook-python/emr-7.0.0-20231211`
- `notebook-python/emr-7.0.0-spark-rapids-latest`
- `notebook-python/emr-7.0.0-spark-rapids-20231211`
- `notebook-python/emr-7.0.0-java11-latest`

- `notebook-python/emr-7.0.0-java11-20231211`
- `notebook-python/emr-7.0.0-java8-latest`
- `notebook-python/emr-7.0.0-java8-20231211`
- `notebook-python/emr-7.0.0-spark-rapids-java8-latest`
- `notebook-python/emr-7.0.0-spark-rapids-java8-20231211`

Notas de la versión

Notas de la versión de Amazon EMR en EKS 7.0.0

- Aplicaciones compatibles: AWS SDK para Java 2.20.160-amzn-0 and 1.12.595, Apache Spark 3.5.0-amzn-0, Apache Flink 1.18.0-amzn-0, Flink Operator 1.6.1, Apache Hudi 0.14.0-amzn-1, Apache Iceberg 1.4.2-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Componentes compatibles: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Clasificaciones de configuración compatibles

Para usar con y: [StartJobRun CreateManagedEndpoint](#) APIs

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
<code>spark-defaults</code>	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .

Clasificaciones	Descripciones
<code>spark-log4j</code>	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
<code>emr-job-submitter</code>	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
<code>jeg-config</code>	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
<code>jupyter-kernel-overrides</code>	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 7.0 de Amazon EMR en EKS.

- Actualizaciones de la aplicación: las actualizaciones de la aplicación de Amazon EMR en EKS 7.0.0 incluyen Spark 3.5, Flink 1.18 y [Flink Operator](#) 1.6.1.
- Ajuste automático de los parámetros del escalador automático de Flink: es posible que los parámetros predeterminados que el escalador automático de Flink utiliza para sus cálculos de escalado no sean el valor óptimo para un trabajo determinado. Amazon EMR en EKS 7.0.0 utiliza las tendencias históricas de métricas capturadas específicas para calcular el parámetro óptimo adaptado al trabajo.

Cambios

Los siguientes cambios se incluyen en la versión 7.0 de Amazon EMR en EKS.

- Amazon Linux 2023: a partir de la versión 7.0.0 de Amazon EMR en EKS, todas las imágenes de contenedores se basan en Amazon Linux 2023.
- Spark utiliza Java 17 como tiempo de ejecución predeterminado: Amazon EMR en EKS 7.0.0 Spark utiliza Java 17 como tiempo de ejecución predeterminado. Si lo necesita, puede cambiar a Java 8 o Java 11 con la etiqueta de versión correspondiente, tal y como se indica en la lista [Versiones 7.0 de Amazon EMR en EKS](#).

emr-7.0.0-latest

Notas de la versión: actualmente, `emr-7.0.0-latest` apunta a `emr-7.0.0-2024321`.

Regiones: `emr-7.0.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.0.0:latest`

emr-7.0.0-2024321

Notas de la versión: `7.0.0-2024321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-7.0.0-2024321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.0.0:2024321`

emr-7.0.0-20231211

Notas de la versión: `7.0.0-20231211` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.0.0 (Spark).

Regiones: `emr-7.0.0-20231211` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.0.0:20231211`

`emr-7.0.0-flink-latest`

Notas de la versión: actualmente, `emr-7.0.0-flink-latest` apunta a `emr-7.0.0-flink-2024321`.

Regiones: `emr-7.0.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.0.0-flink:latest`

`emr-7.0.0-flink-2024321`

Notas de la versión: `7.0.0-flink-2024321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-7.0.0-flink-2024321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.0.0-flink:2024321`

`emr-7.0.0-flink-20231211`

Notas de la versión: `7.0.0-flink-20231211` se lanzó en diciembre de 2023. Esta es la versión inicial de Amazon EMR 7.0.0 (Flink).

Regiones: `emr-7.0.0-flink-20231211` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-7.0.0-flink:20231211`

Versiones 6.15.0 de Amazon EMR en EKS

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 6.15.0 de Amazon EMR en general, consulte Amazon EMR 6.15.0 en la Guía de versiones de Amazon [EMR](#).

Versiones 6.15 de Amazon EMR en EKS

Las siguientes versiones 6.15.0 de Amazon EMR están disponibles para Amazon EMR en EKS. Seleccione una versión específica de emr-6.15.0-XXXX para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

Flink releases

Las siguientes versiones 6.15.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Flink.

- [emr-6.15.0-flink-latest](#)
- [emr-6.15.0-flink-20240105](#)
- [emr-6.15.0-flink-20231109](#)

Spark releases

Las siguientes versiones 6.15.0 de Amazon EMR están disponibles para Amazon EMR en EKS cuando ejecuta aplicaciones de Spark.

- [emr-6.15.0-latest](#)
- [emr-6.15.0-20231109](#)
- emr-6.15.0-spark-rapids-latest
- emr-6.15.0-spark-rapids-20231109
- emr-6.15.0-java11-latest
- emr-6.15.0-java11-20231109
- emr-6.15.0-java17-latest
- emr-6.15.0-java17-20231109
- emr-6.15.0-java17-al2023-latest

- `emr-6.15.0-java17-al2023-20231109`
- `emr-6.15.0-spark-rapids-java17-latest`
- `emr-6.15.0-spark-rapids-java17-20231109`
- `emr-6.15.0-spark-rapids-java17-al2023-latest`
- `emr-6.15.0-spark-rapids-java17-al2023-20231109`
- `notebook-spark/emr-6.15.0-latest`
- `notebook-spark/emr-6.15.0-20231109`
- `notebook-spark/emr-6.15.0-spark-rapids-latest`
- `notebook-spark/emr-6.15.0-spark-rapids-20231109`
- `notebook-spark/emr-6.15.0-java11-latest`
- `notebook-spark/emr-6.15.0-java11-20231109`
- `notebook-spark/emr-6.15.0-java17-latest`
- `notebook-spark/emr-6.15.0-java17-20231109`
- `notebook-spark/emr-6.15.0-java17-al2023-latest`
- `notebook-spark/emr-6.15.0-java17-al2023-20231109`
- `notebook-python/emr-6.15.0-latest`
- `notebook-python/emr-6.15.0-20231109`
- `notebook-python/emr-6.15.0-spark-rapids-latest`
- `notebook-python/emr-6.15.0-spark-rapids-20231109`
- `notebook-python/emr-6.15.0-java11-latest`
- `notebook-python/emr-6.15.0-java11-20231109`
- `notebook-python/emr-6.15.0-java17-latest`
- `notebook-python/emr-6.15.0-java17-20231109`
- `notebook-python/emr-6.15.0-java17-al2023-latest`
- `notebook-python/emr-6.15.0-java17-al2023-20231109`

Notas de la versión

Notas de la versión de Amazon EMR en EKS 6.15.0

- Aplicaciones compatibles: AWS SDK para Java 1.12.569, Apache Spark 3.4.1-amzn-2, Apache Flink 1.17.1-amzn-1, Apache Hudi 0.14.0-amzn-0, Apache Iceberg 1.4.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.08.01-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

Para usar con y: [StartJobRun CreateManagedEndpoint](#) APIs

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 6.15 de Amazon EMR en EKS.

- [Amazon EMR en EKS con Apache Flink](#): con Amazon EMR en EKS 6.15.0, puede ejecutar una aplicación basada en Apache Flink junto con otros tipos de aplicaciones en el mismo clúster de Amazon EKS. Esto ayuda a mejorar la utilización de los recursos y a simplificar la administración de la infraestructura. Puede aprovechar las instancias de spot de una aplicación de Flink con una desactivación rápida y lograr tiempos de reinicio más rápidos con una recuperación detallada y una recuperación local de tareas con Amazon EBS. Las funciones de accesibilidad y supervisión incluyen la posibilidad de lanzar una aplicación Flink con tarros almacenados en Amazon S3, el acceso al catálogo de datos de AWS Glue, la integración de la supervisión con Amazon S3 y Amazon CloudWatch y la rotación de registros de contenedores.

emr-6.15.0-latest

Notas de la versión: actualmente, `emr-6.15.0-latest` apunta a `emr-6.15.0-20240105`.

Regiones: `emr-6.15.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.15.0:latest`

`emr-6.15.0-20240105`

Notas de la versión: `6.15.0-20240105` se lanzó el 17 de enero de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.15.0-20240105` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.15.0:20240105`

`emr-6.15.0-20231109`

Notas de la versión: `6.15.0-20231109` se lanzó el 17 de noviembre de 2023. Esta es la versión inicial de Amazon EMR 6.15.0.

Regiones: `emr-6.15.0-20231109` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.15.0:20231109`

`emr-6.15.0-flink-latest`

Notas de la versión: actualmente, `emr-6.15.0-flink-latest` apunta a `emr-6.15.0-flink-20240105`.

Regiones: `emr-6.15.0-flink-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.15.0-flink:latest`

`emr-6.15.0-flink-20240105`

Notas de la versión: `6.15.0-flink-20240105` se lanzó el 17 de enero de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.15.0-flink-20240105` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.15.0-flink:20240105`

`emr-6.15.0-flink-20231109`

Notas de la versión: `6.15.0-flink-20231109` se lanzó el 17 de noviembre de 2023. Esta es la versión inicial de Amazon EMR 6.15.0.

Regiones: `emr-6.15.0-flink-20231109` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.15.0-flink:20231109`

Versiones de Amazon EMR en EKS 6.14.0

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 6.14.0 de Amazon EMR en general, consulte Amazon EMR 6.14.0 en la Guía de versiones de Amazon [EMR](#).

Versiones de Amazon EMR en EKS 6.14

Las siguientes versiones de Amazon EMR 6.14.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.14.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.14.0-latest](#)
- [emr-6.14.0-20231005](#)
- `emr-6.14.0-spark-rapids-latest`
- `emr-6.14.0-spark-rapids-20231005`
- `emr-6.14.0-java11-latest`
- `emr-6.14.0-java11-20231005`
- `emr-6.14.0-java17-latest`

- emr-6.14.0-java17-20231005
- emr-6.14.0-java17-al2023-latest
- emr-6.14.0-java17-al2023-20231005
- emr-6.14.0-spark-rapids-java17-latest
- emr-6.14.0-spark-rapids-java17-20231005
- emr-6.14.0-spark-rapids-java17-al2023-latest
- emr-6.14.0-spark-rapids-java17-al2023-20231005
- notebook-spark/emr-6.14.0-latest
- notebook-spark/emr-6.14.0-20231005
- notebook-spark/emr-6.14.0-spark-rapids-latest
- notebook-spark/emr-6.14.0-spark-rapids-20231005
- notebook-spark/emr-6.14.0-java11-latest
- notebook-spark/emr-6.14.0-java11-20231005
- notebook-spark/emr-6.14.0-java17-latest
- notebook-spark/emr-6.14.0-java17-20231005
- notebook-spark/emr-6.14.0-java17-al2023-latest
- notebook-spark/emr-6.14.0-java17-al2023-20231005
- notebook-python/emr-6.14.0-latest
- notebook-python/emr-6.14.0-20231005
- notebook-python/emr-6.14.0-spark-rapids-latest
- notebook-python/emr-6.14.0-spark-rapids-20231005
- notebook-python/emr-6.14.0-java11-latest
- notebook-python/emr-6.14.0-java11-20231005
- notebook-python/emr-6.14.0-java17-latest
- notebook-python/emr-6.14.0-java17-20231005
- notebook-python/emr-6.14.0-java17-al2023-latest
- notebook-python/emr-6.14.0-java17-al2023-20231005

Notas de la versión

Notas de la versión de Amazon EMR en EKS 6.14.0

- Aplicaciones compatibles - AWS SDK para Java 1.12.543, Apache Spark 3.4.1-amzn-1, Apache Hudi 0.13.1-amzn-2, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-2, Jupyter Enterprise Gateway 2.7.0
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

[StartJobRun](#) Para [CreateManagedEndpoint](#) APIs usar con y:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 6.14 de Amazon EMR en EKS.

- Compatibilidad con [Apache Livy](#): Amazon EMR en EKS ahora admite Apache Livy con `spark-submit`.

emr-6.14.0-latest

Notas de la versión: actualmente, `emr-6.14.0-latest` apunta a `emr-6.14.0-20231005`.

Regiones: `emr-6.14.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.14.0:latest`

emr-6.14.0-20231005

Notas de la versión: `6.14.0-20231005` se lanzó el 17 de octubre de 2023. Esta es la versión inicial de Amazon EMR 6.14.0.

Regiones: `emr-6.14.0-20231005` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.14.0:20231005`

Versiones de Amazon EMR en EKS 6.13.0

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 6.13.0 de Amazon EMR en general, consulte Amazon EMR 6.13.0 en la Guía de versiones de Amazon [EMR](#).

Versiones de Amazon EMR en EKS 6.13

Las siguientes versiones de Amazon EMR 6.13.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.13.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.13.0-latest](#)
- [emr-6.13.0-20230814](#)
- `emr-6.13.0-spark-rapids-latest`
- `emr-6.13.0-spark-rapids-20230814`
- `emr-6.13.0-java11-latest`
- `emr-6.13.0-java11-20230814`
- `emr-6.13.0-java17-latest`
- `emr-6.13.0-java17-20230814`
- `emr-6.13.0-java17-al2023-latest`
- `emr-6.13.0-java17-al2023-20230814`
- `emr-6.13.0-spark-rapids-java17-latest`
- `emr-6.13.0-spark-rapids-java17-20230814`
- `emr-6.13.0-spark-rapids-java17-al2023-latest`
- `emr-6.13.0-spark-rapids-java17-al2023-20230814`
- `notebook-spark/emr-6.13.0-latest`

- notebook-spark/emr-6.13.0-20230814
- notebook-spark/emr-6.13.0-spark-rapids-latest
- notebook-spark/emr-6.13.0-spark-rapids-20230814
- notebook-spark/emr-6.13.0-java11-latest
- notebook-spark/emr-6.13.0-java11-20230814
- notebook-spark/emr-6.13.0-java17-latest
- notebook-spark/emr-6.13.0-java17-20230814
- notebook-spark/emr-6.13.0-java17-al2023-latest
- notebook-spark/emr-6.13.0-java17-al2023-20230814
- notebook-python/emr-6.13.0-latest
- notebook-python/emr-6.13.0-20230814
- notebook-python/emr-6.13.0-spark-rapids-latest
- notebook-python/emr-6.13.0-spark-rapids-20230814
- notebook-python/emr-6.13.0-java11-latest
- notebook-python/emr-6.13.0-java11-20230814
- notebook-python/emr-6.13.0-java17-latest
- notebook-python/emr-6.13.0-java17-20230814
- notebook-python/emr-6.13.0-java17-al2023-latest
- notebook-python/emr-6.13.0-java17-al2023-20230814

Notas de la versión

Notas de la versión de Amazon EMR en EKS 6.13.0

- Aplicaciones compatibles - AWS SDK para Java 1.12.513, Apache Spark 3.4.1-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-1, Jupyter Enterprise Gateway 2.6.0.amzn
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

[StartJobRun](#) Para [CreateManagedEndpoint](#) API usar con y:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .
emr-job-submitter	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 6.13 de Amazon EMR en EKS.

- **Amazon Linux 2023:** con Amazon EMR en EKS 6.13 y versiones posteriores, puede lanzar Spark con AL2 023 como sistema operativo junto con el tiempo de ejecución de Java 17. Para ello, utilice la etiqueta de versión que incluya `al2023` en su nombre. Por ejemplo: `emr-6.13.0-java17-al2023-latest`. Le recomendamos que valide y ejecute pruebas de rendimiento antes de migrar sus cargas de trabajo de producción a AL2 023 y Java 17.
- [Amazon EMR en EKS con Apache Flink](#) (versión preliminar pública): las versiones 6.13 y posteriores de Amazon EMR en EKS son compatibles con Apache Flink, disponible en la versión preliminar pública. Con este lanzamiento, puede ejecutar su aplicación basada en Apache Flink junto con otros tipos de aplicaciones en el mismo clúster de Amazon EKS. Esto ayuda a mejorar la utilización de los recursos y a simplificar la administración de la infraestructura. Si ya ejecuta marcos de macrodatos en Amazon EKS, ahora puede dejar que Amazon EMR automatice el aprovisionamiento y la administración.

emr-6.13.0-latest

Notas de la versión: actualmente, `emr-6.13.0-latest` apunta a `emr-6.13.0-20230814`.

Regiones: `emr-6.13.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.13.0:latest`

emr-6.13.0-20230814

Notas de la versión: `6.13.0-20230814` se lanzó el 7 de septiembre de 2023. Esta es la versión inicial de Amazon EMR 6.13.0.

Regiones: `emr-6.13.0-20230814` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.13.0:20230814`

Versiones de Amazon EMR en EKS 6.12.0

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 6.12.0 de Amazon EMR en general, consulte Amazon EMR 6.12.0 en la Guía de versiones de Amazon [EMR](#).

Versiones de Amazon EMR en EKS 6.12

Las siguientes versiones de Amazon EMR 6.12.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.12.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.12.0-latest](#)
- [emr-6.12.0-20240321](#)
- [emr-6.12.0-20230701](#)
- `emr-6.12.0-spark-rapids-latest`
- `emr-6.12.0-spark-rapids-20230701`
- `emr-6.12.0-java11-latest`
- `emr-6.12.0-java11-20230701`
- `emr-6.12.0-java17-latest`
- `emr-6.12.0-java17-20230701`
- `emr-6.12.0-spark-rapids-java 17 - más reciente`
- `emr-6.12.0-spark-rapids-java 17-20230701`
- `notebook-spark/emr-6.12.0-latest`
- `notebook-spark/emr-6.12.0-20230701`
- `cuaderno: spark/emr-6.12.0-spark-rapids-latest`
- `notebook-spark/emr-6.12.0-spark-rapids-20230701`
- `notebook-python/emr-6.12.0-latest`

- notebook-python/emr-6.12.0-20230701
- cuaderno: python/emr-6.12.0- spark-rapids-latest
- notebook-python/emr-6.12.0-spark-rapids-20230701

Notas de la versión

Notas de la versión de Amazon EMR en EKS 6.12.0

- Aplicaciones compatibles - AWS SDK para Java 1.12.490, Apache Spark 3.4.0-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles

[StartJobRun](#) [CreateManagedEndpoint](#) APIs Para usar con y:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .
spark-defaults	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
spark-log4j	Cambia los valores en el archivo de Spark <code>log4j2.properties</code> .

Clasificaciones	Descripciones
<code>emr-job-submitter</code>	Configuración del pod de remitente de trabajos .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
<code>jeg-config</code>	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
<code>jupyter-kernel-overrides</code>	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 6.12 de Amazon EMR en EKS.

- **Java 17:** con Amazon EMR en EKS 6.12 y versiones posteriores, puede lanzar Spark con el tiempo de ejecución de Java 17. Para ello, pase `emr-6.12.0-java17-latest` como una etiqueta de versión. Le recomendamos que valide y ejecute pruebas de rendimiento antes de trasladar las cargas de trabajo de producción de versiones anteriores de la imagen de Java a la imagen de Java 17.

emr-6.12.0-latest

Notas de la versión: actualmente, `emr-6.12.0-latest` apunta a `emr-6.12.0-20240321`.

Regiones: `emr-6.12.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.12.0:latest`

`emr-6.12.0-20240321`

Notas de la versión: `6.12.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.12.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.12.0:20240321`

`emr-6.12.0-20230701`

Notas de la versión: `6.12.0-20230701` se lanzó el 1 de julio de 2023. Esta es la versión inicial de Amazon EMR 6.12.0.

Regiones: `emr-6.12.0-20230701` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.12.0:20230701`

Versiones de Amazon EMR en EKS 6.11.0

Esta página describe la funcionalidad nueva y actualizada de Amazon EMR que es específica de la implementación de Amazon EMR en EKS. Para obtener más información sobre Amazon EMR que se ejecuta en Amazon EC2 y sobre la versión 6.11.0 de Amazon EMR en general, consulte Amazon EMR 6.11.0 en la Guía de versiones de Amazon [EMR](#).

Versiones de Amazon EMR en EKS 6.11

Las siguientes versiones de Amazon EMR 6.11.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.11.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.11.0-latest](#)
- [emr-6.11.0-20230905](#)
- [emr-6.11.0-20230509](#)
- `emr-6.11.0-spark-rapids-latest`
- `emr-6.11.0-spark-rapids-20230509`
- `emr-6.11.0-java11-latest`
- `emr-6.11.0-java11-20230509`
- `notebook-spark/emr-6.11.0-latest`
- `notebook-spark/emr-6.11.0-20230509`
- `notebook-python/emr-6.11.0-latest`
- `notebook-python/emr-6.11.0-20230509`

Notas de la versión

Notas de la versión de Amazon EMR en EKS 6.11.0

- Aplicaciones compatibles: AWS SDK para Java 1.12.446, Apache Spark 3.3.2-amzn-0, Apache Hudi 0.13.0-amzn-0, Apache Iceberg 1.2.0-amzn-0, Delta 2.2.0, Apache Spark RAPIDS 23.02.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Componentes compatibles: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Clasificaciones de configuración compatibles

[StartJobRun](#) [CreateManagedEndpoint](#) APIs Para usar con y:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo de Hadoop <code>core-site.xml</code> .
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambia los valores en el archivo de Spark <code>metrics.properties</code> .

Clasificaciones	Descripciones
<code>spark-defaults</code>	Cambia los valores en el archivo de Spark <code>spark-defaults.conf</code> .
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo de Spark <code>hive-site.xml</code> .
<code>spark-log4j</code>	Cambia los valores en el archivo de Spark <code>log4j.properties</code> .

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
<code>jeg-config</code>	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
<code>jupyter-kernel-overrides</code>	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

Las siguientes características se incluyen en la versión 6.11 de Amazon EMR en EKS.

- [Imagen base de Amazon EMR en EKS en Amazon ECR Public Gallery](#): si utiliza la capacidad de [imagen personalizada](#), nuestra imagen base proporciona los jars, la configuración y las bibliotecas

esenciales para interactuar con Amazon EMR en EKS. Ahora puede encontrar la imagen base en [Amazon ECR Public Gallery](#).

- [Rotación de registros de contenedores de Spark](#): Amazon EMR en EKS 6.11 es compatible con la rotación de registros de contenedores de Spark. Puede activar la capacidad con `containerLogRotationConfiguration` desde la operación `MonitoringConfiguration` de la API `StartJobRun`. Puede configurar `rotationSize` y `maxFilestoKeep` para especificar el número y el tamaño de los archivos de registro que desea que Amazon EMR en EKS mantenga en los pods ejecutores y controladores de Spark. Para obtener más información, consulte [Uso de la rotación de los registros de contenedores de Spark](#).
- Compatibilidad con Volcano en el operador de Spark y `spark-submit`: Amazon EMR en EKS 6.11 admite la ejecución de trabajos de Spark con Volcano como programador personalizado de Kubernetes en el [operador de Spark](#) y [spark-submit](#). Puede utilizar características como la planificación por grupos, la administración de colas, la prevención y la programación equitativa para lograr un alto rendimiento de la programación y una capacidad optimizada. Para obtener más información, consulte [Uso de Volcano como programador personalizado para Apache Spark en Amazon EMR en EKS](#).

emr-6.11.0-latest

Notas de la versión: actualmente, `emr-6.11.0-latest` apunta a `emr-20230905`.

Regiones: `emr-6.11.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.11.0:latest`

emr-6.11.0-20230905

Notas de la versión: `6.11.0-20230905` se lanzó el 29 de septiembre de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.11.0-20230509` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.11.0:20230509`

emr-6.11.0-20230509

Notas de la versión: 6.11.0-20230509 se lanzó el 9 de mayo de 2023. Esta es la versión inicial de Amazon EMR 6.11.0.

Regiones: `emr-6.11.0-20230509` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.11.0:20230509`

Versiones de Amazon EMR en EKS 6.10.0

Las siguientes versiones de Amazon EMR 6.10.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.10.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.10.0-latest](#)
- [emr-6.10.0-20230905](#)
- [emr-6.10.0-20230624](#)
- [emr-6.10.0-20230421](#)
- [emr-6.10.0-20230403](#)
- [emr-6.10.0-20230220](#)
- `emr-6.10.0-spark-rapids-latest`
- `emr-6.10.0-spark-rapids-20230624`
- `emr-6.10.0-spark-rapids-20230220`
- `emr-6.10.0-java11-latest`
- `emr-6.10.0-java11-20230624`
- `emr-6.10.0-java11-20230220`
- `notebook-spark/emr-6.10.0-latest`
- `notebook-spark/emr-6.10.0-20230624`
- `notebook-spark/emr-6.10.0-20230220`
- `notebook-python/emr-6.10.0-latest`

- `notebook-python/emr-6.10.0-20230624`
- `notebook-python/emr-6.10.0-20230220`

Notas de la versión de Amazon EMR 6.10.0

- Aplicaciones compatibles: AWS SDK para Java 1.12.397, Spark 3.3.1-amzn-0, Hudi 0.12.2-amzn-0, Iceberg 1.1.0-amzn-0 y Delta 2.2.0.
- Componentes compatibles: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Clasificaciones de configuración compatibles:

[StartJobRun](#) Para [CreateManagedEndpoint](#) APIs usar con y:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambia los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambia los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambia los valores en el archivo <code>log4j.properties</code> de Spark.

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
jeg-config	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

- **Operador de Spark:** con Amazon EMR en EKS 6.10.0 y versiones posteriores, puede usar el operador de Kubernetes para Apache Spark, o el operador de Spark, para implementar y administrar aplicaciones de Spark con el tiempo de ejecución de versiones de Amazon EMR en sus propios clústeres de Amazon EKS. Para obtener más información, consulte [Ejecución de trabajos de Spark con el operador de Spark](#).
- **Java 11:** con Amazon EMR en EKS 6.10 y versiones posteriores, puede lanzar Spark con el tiempo de ejecución de Java 11. Para ello, pase `emr-6.10.0-java11-latest` como una etiqueta de versión. Le recomendamos que valide y ejecute pruebas de rendimiento antes de mover las cargas de trabajo de producción de la imagen de Java 8 a la imagen de Java 11.
- Para la integración de Amazon Redshift con Apache Spark, Amazon EMR en EKS 6.10.0 elimina la dependencia de `minimal-json.jar` y agrega automáticamente los jars de `spark-redshift` a la ruta de clases del ejecutor de Spark: `spark-redshift.jar`, `spark-avro.jar` y `RedshiftJDBC.jar`.

Cambios

- El confirmador EMRFS optimizado para S3 ahora está habilitado de forma predeterminada para formatos Parquet, ORC y basados en texto (incluidos CSV y JSON).

emr-6.10.0-latest

Notas de la versión: actualmente, `emr-6.10.0-latest` apunta a `emr-6.10.0-20230905`.

Regiones: `emr-6.10.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.10.0:latest`

emr-6.10.0-20230905

Notas de la versión: `6.10.0-20230905` se lanzó el 29 de septiembre de 2023. En comparación con la versión anterior, esta versión se ha actualizado con paquetes de Amazon Linux actualizados recientemente y se le han aplicado correcciones críticas.

Regiones: `emr-6.10.0-20230905` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.10.0:20230905`

emr-6.10.0-20230624

Notas de la versión: `6.10.0-20230624` se lanzó el 7 de julio de 2023. En comparación con la versión anterior, esta versión se ha actualizado con paquetes de Amazon Linux actualizados recientemente y se le han aplicado correcciones críticas.

Regiones: `emr-6.10.0-20230624` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.10.0:20230624`

emr-6.10.0-20230421

Notas de la versión: `6.10.0-20230421` se lanzó el 28 de abril de 2023. En comparación con la versión anterior, esta versión se ha actualizado con paquetes de Amazon Linux actualizados recientemente y se le han aplicado correcciones críticas.

Regiones: `emr-6.10.0-20230421` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.10.0:20230421`

`emr-6.10.0-20230403`

Notas de la versión: `6.10.0-20230403` se lanzó el 12 de abril de 2023. En comparación con la versión anterior, esta versión se ha actualizado con paquetes de Amazon Linux actualizados recientemente y se le han aplicado correcciones críticas.

Regiones: `emr-6.10.0-20230403` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.10.0:20230403`

`emr-6.10.0-20230220`

Notas de la versión: `emr-6.10.0-20230220` se lanzó el 20 de febrero de 2023. Esta es la versión inicial de Amazon EMR 6.10.0.

Regiones: `emr-6.10.0-20230220` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.10.0:20230220`

Versiones de Amazon EMR en EKS 6.9.0

Las siguientes versiones de Amazon EMR 6.9.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.9.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.9.0-latest](#)
- [emr-6.9.0-20230905](#)
- [emr-6.9.0-20230624](#)
- [emr-6.9.0-20221108](#)
- `emr-6.9.0-spark-rapids-latest`

- `emr-6.9.0-spark-rapids-20230624`
- `emr-6.9.0-spark-rapids-20221108`
- `notebook-spark/emr-6.9.0-latest`
- `notebook-spark/emr-6.9.0-20230624`
- `notebook-spark/emr-6.9.0-20221108`
- `notebook-python/emr-6.9.0-latest`
- `notebook-python/emr-6.9.0-20230624`
- `notebook-python/emr-6.9.0-20221108`

Notas de la versión de Amazon EMR 6.9.0

- Aplicaciones compatibles: AWS SDK para Java 1.12.331, Spark 3.3.0-amzn-1, Hudi 0.12.1-amzn-0, Iceberg 0.14.1-amzn-0 y Delta 2.1.0.
- Componentes compatibles: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Clasificaciones de configuración compatibles:

[StartJobRun](#) Para [CreateManagedEndpoint](#) APIs usar con y:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.

Clasificaciones	Descripciones
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Para su uso específico con [CreateManagedEndpoint](#) APIs:

Clasificaciones	Descripciones
<code>jeg-config</code>	Cambia los valores en el archivo <code>jupyter_enterprise_gateway_config.py</code> de Jupyter Enterprise Gateway.
<code>jupyter-kernel-overrides</code>	Cambia el valor de la imagen del kernel en el archivo de especificaciones del kernel de Jupyter.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

- Nvidia RAPIDS Accelerator para Apache Spark - Amazon EMR en EKS para acelerar Spark mediante tipos de instancias de unidades de procesamiento EC2 gráfico (GPU). Para usar la imagen de Spark con RAPIDS Accelerator, especifique la etiqueta de lanzamiento como `emr-6.9.0-spark-rapids-latest`. Visite la [página de documentación](#) para obtener más información.
- Conector Spark-Redshift: 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 lean 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 en EKS](#).
- Delta Lake: [Delta Lake](#) es un formato de almacenamiento de código abierto que permite crear lagos de datos con coherencia transaccional, una definición coherente de los conjuntos de datos,

cambios en la evolución de los esquemas y compatibilidad con las mutaciones de datos. Visite [Uso de Delta Lake](#) para obtener más información.

- Modificar PySpark parámetros - Los puntos finales interactivos ahora admiten la modificación de los parámetros de Spark asociados a PySpark las sesiones en el cuaderno Jupyter de EMR Studio. Visite [Modificación de los parámetros de la PySpark sesión](#) para obtener más información.

Problemas resueltos

- Cuando utiliza el conector de DynamoDB con Spark en las versiones 6.6.0, 6.7.0 y 6.8.0 de Amazon EMR, todas las lecturas de la tabla devuelven un resultado vacío, aunque la división de entrada haga referencia a datos que no están vacíos. La versión 6.9.0 de Amazon EMR corrige este problema.
- Amazon EMR en EKS 6.8.0 rellena incorrectamente el hash de compilación en los metadatos de los archivos Parquet generados con [Apache Spark](#). Este problema puede provocar un error en las herramientas que analizan la cadena de versión de metadatos de los archivos Parquet generados por Amazon EMR en EKS 6.8.0.

Problema conocido

- Si utiliza la integración de Amazon Redshift para Apache Spark y tiene un valor de time, timetz, timestamp o timestampz 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-6.9.0-latest

Notas de la versión: actualmente, `emr-6.9.0-latest` apunta a `emr-6.9.0-20230905`.

Regiones: `emr-6.9.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.9.0:latest`

emr-6.9.0-20230905

Notas de la versión: `emr-6.9.0-20230905`. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.9.0-20230905` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.9.0:20230905`

emr-6.9.0-20230624

Notas de la versión: `emr-6.9.0-20230624` se lanzó el 7 de julio de 2023.

Regiones: `emr-6.9.0-20230624` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.9.0:20230624`

emr-6.9.0-20221108

Notas de la versión: `emr-6.9.0-20221108` se lanzó el 8 de diciembre de 2022. Esta es la versión inicial de Amazon EMR 6.9.0.

Regiones: `emr-6.9.0-20221108` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.9.0:20221108`

Versiones de Amazon EMR en EKS 6.8.0

Las siguientes versiones de Amazon EMR 6.8.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.8.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.8.0-latest](#)
- [emr-6.8.0-20230905](#)

- [emr-6.8.0-20230624](#)
- [emr-6.8.0-20221219](#)
- [emr-6.8.0-20220802](#)

Notas de la versión de Amazon EMR 6.8.0

- Aplicaciones compatibles: AWS SDK para Java 1.12.170, Spark 3.3.0-amzn-0, Hudi 0.11.1-amzn-0 e Iceberg 0.14.0-amzn-0.
- Componentes compatibles: aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-s3-select, emrfs, hadoop-client, hudi, hudi-spark, iceberg, spark-kubernetes.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo core-site.xml de Hadoop.
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambiar los valores en el archivo metrics.properties de Spark.
spark-defaults	Cambiar los valores en el archivo spark-defaults.conf de Spark.
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo hive-site.xml de Spark.
spark-log4j	Cambiar los valores en el archivo log4j.properties de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, spark-hive-site.xml. Para obtener más información, consulte [Configuración de aplicaciones](#).

Características notables

- Spark 3.3.0: Amazon EMR en EKS 6.8 incluye Spark 3.3.0, que admite el uso de etiquetas de selector de nodos independientes para los pods ejecutores de controladores de Spark. Estas nuevas etiquetas permiten definir los tipos de nodos para los módulos del controlador y del ejecutor por separado en la API, sin utilizar plantillas de módulos. `StartJobRun`
- Propiedad del selector de nodos del controlador: `spark.kubernetes.driver.node.selector.[labelKey]`
- Propiedad del selector de nodos del ejecutor: `spark.kubernetes.executor.node.selector.[labelKey]`
- Mensaje de error de trabajo mejorado: esta versión presenta la configuración `spark.stage.extraDetailsOnFetchFailures.enabled` y `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` para hacer un seguimiento de los errores en las tareas debidos al código del usuario. Estos detalles se utilizarán para mejorar el mensaje de error que se muestra en el registro del controlador cuando se cancela una etapa debido a un error en la recuperación aleatoria.

Nombre de la propiedad	Valor predeterminado	Significado	Desde la versión
<code>spark.stage.extraDetailsOnFetchFailures.enabled</code>	false	Si se establece en true, esta propiedad se utiliza para mejorar el mensaje de error del trabajo que aparece en el registro del controlador cuando se interrumpe una etapa debido a un error de captura aleatoria. De forma predeterminada, se hace un seguimiento de los cinco últimos errores de tareas	emr-6.8

Nombre de la propiedad	Valor predeterminado	Significado	Desde la versión
		causados por el código del usuario y el mensaje de error se adjunta a los registros de controlador. Para aumentar el número de errores de tareas con excepciones de usuario para hacer un seguimiento, consulte la configuración <code>spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude</code>.	

Nombre de la propiedad	Valor predeterminado	Significado	Desde la versión
<code>spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude</code>	5	Número de errores en las tareas que se deben rastrear por etapa e intento. Esta propiedad se utiliza para mejorar el mensaje de error de un trabajo, ya que las excepciones de usuario se muestran en el registro del controlador cuando se interrumpe una etapa debido a un error de captura aleatoria. Esta propiedad solo funciona si <code>Config</code> es <code>spark.stage.extraDetailsOnFetchFailures.enabled</code> tiene el valor <code>true</code>.	emr-6.8

Para obtener más información, consulte la [documentación de configuración de Apache Spark](#).

Problema conocido

- Amazon EMR en EKS 6.8.0 rellena incorrectamente el hash de compilación en los metadatos de los archivos Parquet generados con [Apache Spark](#). Este problema puede provocar un error en las herramientas que analizan la cadena de versión de metadatos de los archivos Parquet generados por Amazon EMR en EKS 6.8.0. Los clientes que analicen la cadena de versión a partir de los metadatos de Parquet y dependan del hash de compilación deberían cambiar a una versión diferente de Amazon EMR y volver a escribir el archivo.

Problema resuelto

- Interrupción de la capacidad del kernel para kernels de PySpark: las cargas de trabajo interactivas en curso que se activan al ejecutar celdas en un cuaderno se pueden detener mediante la capacidad `Interrupt Kernel`. Se ha introducido una solución para que esta característica funcione para los kernels de PySpark. También está disponible en código abierto en [Changes for handling interrupts for PySpark Kubernetes Kernel #1115](#).

emr-6.8.0-latest

Notas de la versión: actualmente, `emr-6.8.0-latest` apunta a `emr-6.8.0-20230624`.

Regiones: `emr-6.8.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.8.0:latest`

emr-6.8.0-20230905

Notas de la versión: `emr-6.8.0-20230905` se lanzó el 29 de septiembre de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.8.0-20230905` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.8.0:20230905`

emr-6.8.0-20230624

Notas de la versión: `emr-6.8.0-20230624` se lanzó el 7 de julio de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.8.0-20230624` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.8.0:20230624`

emr-6.8.0-20221219

Notas de la versión: `emr-6.8.0-20221219` se lanzó el 19 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.8.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.8.0:20221219`

emr-6.8.0-20220802

Notas de la versión: `emr-6.8.0-20220802` se lanzó el 27 de septiembre de 2022. Esta es la versión inicial de Amazon EMR 6.8.0.

Regiones: `emr-6.8.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.8.0:20220802`

Versiones de Amazon EMR en EKS 6.7.0

Las siguientes versiones de Amazon EMR 6.7.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.7.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.7.0-latest](#)
- [emr-6.7.0-20240321](#)
- [emr-6.7.0-20230624](#)
- [emr-6.7.0-20221219](#)
- [emr-6.7.0-20220630](#)

Notas de la versión de Amazon EMR 6.7.0

- Aplicaciones compatibles: Spark 3.2.1-amzn-0, Jupyter Enterprise Gateway 2.6, Hudi 0.11-amzn-0, Iceberg 0.13.1.

- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Con la actualización a JEG 2.6, la administración del kernel ahora es asíncrona, lo que significa que JEG no bloquea las transacciones cuando se está lanzando el kernel. Esto mejora considerablemente la experiencia del usuario al proporcionar lo siguiente:
 - capacidad de ejecutar comandos en los cuadernos que ya se estén ejecutando cuando se estén ejecutando otros lanzamientos del kernel
 - capacidad de lanzar varios kernels de manera simultánea sin afectar a los kernels que ya se están ejecutando
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambia los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambia los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambia los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración para la aplicación como, por ejemplo, `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Problemas resueltos

- Amazon EMR en EKS 6.7 corrige un problema en la versión 6.6 al utilizar la funcionalidad de plantillas de pods de Apache Spark con puntos de conexión interactivos. El problema estaba presente en las versiones 6.4, 6.5 y 6.6 de Amazon EMR en EKS. Ahora puede usar plantillas de pods para definir cómo se inician sus pods controladores y ejecutores de Spark cuando utiliza puntos de conexión interactivos para ejecutar análisis interactivos.
- En versiones anteriores de Amazon EMR en EKS, Jupyter Enterprise Gateway bloqueaba las transacciones cuando el lanzamiento del kernel estaba en curso, lo que impedía la ejecución de las sesiones de cuaderno que se estaban ejecutando. Ahora puede ejecutar comandos en los cuadernos que estén en ejecución cuando se estén lanzando otros kernels. También puede lanzar varios kernels simultáneamente sin correr el riesgo de perder la conectividad con los kernels que ya se estén ejecutando.

emr-6.7.0-latest

Notas de la versión: actualmente, `emr-6.7.0-latest` apunta a `emr-6.7.0-20240321`.

Regiones: `emr-6.7.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.7.0:latest`

emr-6.7.0-20240321

Notas de la versión: `emr-6.7.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.7.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.7.0:20240321`

emr-6.7.0-20230624

Notas de la versión: `emr-6.7.0-20230624` se lanzó el 7 de julio de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.7.0-20230624` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.7.0:20230624`

emr-6.7.0-20221219

Notas de la versión: `emr-6.7.0-20221219` se lanzó el 19 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.7.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.7.0:20221219`

emr-6.7.0-20220630

Notas de la versión: `emr-6.7.0-20220630` se lanzó el 12 de julio de 2022. Esta es la versión inicial de Amazon EMR 6.7.0.

Regiones: `emr-6.7.0-20220630` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.7.0:20220630`

Versiones de Amazon EMR en EKS 6.6.0

Las siguientes versiones de Amazon EMR 6.6.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.6.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.6.0-latest](#)
- [emr-6.6.0-20240321](#)
- [emr-6.6.0-20230624](#)
- [emr-6.6.0-20221219](#)
- [emr-6.6.0-20220411](#)

Notas de la versión de Amazon EMR 6.6.0

- Aplicaciones compatibles: Spark 3.2.0-amzn-0, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública), Hudi 0.10.1-amzn-0, Iceberg 0.13.1.
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `.xml.spark-hive-site`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Problema conocido

- La funcionalidad de la plantilla de pod de Spark con puntos de conexión interactivos no funciona en las versiones 6.4, 6.5 y 6.6 de Amazon EMR en EKS.

Problema resuelto

- Los registros interactivos de los puntos de conexión se cargan en CloudWatch y S3.

emr-6.6.0-latest

Notas de la versión: actualmente, `emr-6.6.0-latest` apunta a `emr-6.6.0-20240321`.

Regiones: `emr-6.6.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.6.0:latest`

emr-6.6.0-20240321

Notas de la versión: `emr-6.6.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.6.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.6.0:20240321`

emr-6.6.0-20230624

Notas de la versión: `emr-6.6.0-20230624` se lanzó el 27 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.6.0-20230624` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.6.0:20230624`

emr-6.6.0-20221219

Notas de la versión: `emr-6.6.0-20221219` se lanzó el 27 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.6.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.6.0:20221219`

emr-6.6.0-20220411

Notas de la versión: `emr-6.6.0-20220411` se lanzó el 20 de mayo de 2022. Esta es la versión inicial de Amazon EMR 6.6.0.

Regiones: `emr-6.6.0-20220411` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.6.0:20220411`

Versiones de Amazon EMR en EKS 6.5.0

Las siguientes versiones de Amazon EMR 6.5.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.5.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.5.0-latest](#)
- [emr-6.5.0-20240321](#)
- [emr-6.5.0-20221219](#)
- [emr-6.5.0-20220802](#)
- [emr-6.5.0-20211119](#)

Notas de la versión de Amazon EMR 6.5.0

- Aplicaciones compatibles: Spark 3.1.2-amzn-1, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública).
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Problema conocido

- La funcionalidad de la plantilla de pod de Spark con puntos de conexión interactivos no funciona en las versiones 6.4 y 6.5 de Amazon EMR en EKS.

emr-6.5.0-latest

Notas de la versión: actualmente, `emr-6.5.0-latest` apunta a `emr-6.5.0-20240321`.

Regiones: `emr-6.5.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.5.0:latest`

emr-6.5.0-20240321

Notas de la versión: `emr-6.5.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.5.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.5.0:20240321`

emr-6.5.0-20221219

Notas de la versión: `emr-6.5.0-20221219` se lanzó el 19 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.5.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.5.0:20221219`

`emr-6.5.0-20220802`

Notas de la versión: `emr-6.5.0-20220802` se lanzó el 24 de agosto de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-6.5.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.5.0:20220802`

`emr-6.5.0-20211119`

Notas de la versión: `emr-6.5.0-20211119` se lanzó el 20 de enero de 2022. Esta es la versión inicial de Amazon EMR 6.5.0.

Regiones: `emr-6.5.0-20211119` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.5.0:20211119`

Versiones de Amazon EMR en EKS 6.4.0

Las siguientes versiones de Amazon EMR 6.4.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.4.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.4.0-latest](#)
- [emr-6.4.0-20240321](#)
- [emr-6.4.0-20221219](#)
- [emr-6.4.0-20210830](#)

Notas de la versión de Amazon EMR 6.4.0

- Aplicaciones compatibles: Spark 3.1.2-amzn-0, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública).
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

Problema conocido

- La funcionalidad de la plantilla de pod de Spark con puntos de conexión interactivos no funciona en la versión 6.4 de Amazon EMR en EKS.

emr-6.4.0-latest

Notas de la versión: actualmente, `emr-6.4.0-latest` apunta a `emr-6.4.0-20240321`.

Regiones: `emr-6.4.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.4.0:latest`

emr-6.4.0-20240321

Notas de la versión: `emr-6.4.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.4.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.4.0:20240321`

emr-6.4.0-20221219

Notas de la versión: `emr-6.4.0-20221219` se lanzó el 27 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente agregados.

Regiones: `emr-6.4.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.4.0:20221219`

emr-6.4.0-20210830

Notas de la versión: `emr-6.4.0-20210830` se lanzó el 9 de diciembre de 2021. Esta es la versión inicial de Amazon EMR 6.4.0.

Regiones: `emr-6.4.0-20210830` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.4.0:20210830`

Versiones de Amazon EMR en EKS 6.3.0

Las siguientes versiones de Amazon EMR 6.3.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.3.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.3.0-latest](#)
- [emr-6.3.0-20240321](#)
- [emr-6.3.0-20220802](#)
- [emr-6.3.0-20211008](#)
- [emr-6.3.0-20210802](#)
- [emr-6.3.0-20210429](#)

Notas de la versión de Amazon EMR 6.3.0

- Nuevas características: a partir de Amazon EMR 6.3.0 en la serie de versiones 6.x, Amazon EMR en EKS es compatible con la característica de plantillas de pods de Spark. También puede activar la característica de rotación del registro de eventos de Spark de Amazon EMR en EKS. Para obtener más información, consulte [Uso de plantillas de pods](#) y [Uso de la rotación del registro de eventos de Spark](#).
- Aplicaciones compatibles: Spark 3.1.1-amzn-0, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública).
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.

Clasificaciones	Descripciones
spark-metrics	Cambiar los valores en el archivo metrics.properties de Spark.
spark-defaults	Cambiar los valores en el archivo spark-defaults.conf de Spark.
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo hive-site.xml de Spark.
spark-log4j	Cambiar los valores en el archivo log4j.properties de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como spark-hive-site.xml. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-6.3.0-latest

Notas de la versión: actualmente, `emr-6.3.0-latest` apunta a `emr-6.3.0-20240321`.

Regiones: `emr-6.3.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.3.0:latest`

emr-6.3.0-20240321

Notas de la versión: `emr-6.3.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.3.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.3.0:20240321`

emr-6.3.0-20220802

Notas de la versión: `emr-6.3.0-20220802` se lanzó el 27 de septiembre de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-6.3.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.3.0:20220802`

emr-6.3.0-20211008

Notas de la versión: `emr-6.3.0-20211008` se lanzó el 9 de diciembre de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.3.0-20211008` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.3.0:20211008`

emr-6.3.0-20210802

Notas de la versión: `emr-6.3.0-20210802` se lanzó el 2 de agosto de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.3.0-20210802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.3.0:20210802`

emr-6.3.0-20210429

Notas de la versión: `emr-6.3.0-20210429` se lanzó el 29 de abril de 2021. Esta es la versión inicial de Amazon EMR 6.3.0.

Regiones: `emr-6.3.0-20210429` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.3.0:20210429`

Versiones de Amazon EMR en EKS 6.2.0

Las siguientes versiones de Amazon EMR 6.2.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-6.2.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-6.2.0-latest](#)
- [emr-6.2.0-20240321](#)
- [emr-6.2.0-20220802](#)
- [emr-6.2.0-20211008](#)
- [emr-6.2.0-20210802](#)
- [emr-6.2.0-20210615](#)
- [emr-6.2.0-20210129](#)
- [emr-6.2.0-20201218](#)
- [emr-6.2.0-20201201](#)

Notas de la versión de Amazon EMR 6.2.0

- Aplicaciones compatibles: Spark 3.0.1-amzn-0, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública).
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.

Clasificaciones	Descripciones
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-6.2.0-latest

Notas de la versión: actualmente, `emr-6.2.0-latest` apunta a `emr-6.2.0-20240321`.

Regiones: `emr-6.2.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.2.0:20240321`

emr-6.2.0-20240321

Notas de la versión: `emr-6.2.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-6.2.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.2.0:20240321`

`emr-6.2.0-20220802`

Notas de la versión: `emr-6.2.0-20220802` se lanzó el 27 de septiembre de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-6.2.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-6.2.0:20220802`

`emr-6.2.0-20211008`

Notas de la versión: `emr-6.2.0-20211008` se lanzó el 9 de diciembre de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.2.0-20211008` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-6.2.0:20211008`

`emr-6.2.0-20210802`

Notas de la versión: `emr-6.2.0-20210802` se lanzó el 2 de agosto de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.2.0-20210802` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-6.2.0:20210802`

emr-6.2.0-20210615

Notas de la versión: `emr-6.2.0-20210615` se lanzó el 15 de junio de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.2.0-20210615` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-6.2.0:20210615`

emr-6.2.0-20210129

Notas de la versión: `emr-6.2.0-20210129` se lanzó el 29 de enero de 2021. En comparación con `emr-6.2.0-20201218`, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.2.0-20210129` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-6.2.0-20210129`

emr-6.2.0-20201218

Notas de la versión: `emr-6.2.0-20201218` se lanzó el 18 de diciembre de 2020. En comparación con `emr-6.2.0-20201201`, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-6.2.0-20201218` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-6.2.0-20201218`

emr-6.2.0-20201201

Notas de la versión: `emr-6.2.0-20201201` se lanzó el 1 de diciembre de 2020. Esta es la versión inicial de Amazon EMR 6.2.0.

Regiones: `emr-6.2.0-20201201` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-6.2.0-20201201`

Versiones de Amazon EMR en EKS 5.36.0

Las siguientes versiones de Amazon EMR 5.36.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-5.36.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-5.36.0-latest](#)
- [emr-5.36.0-20240321](#)
- [emr-5.36.0-20221219](#)
- [emr-5.36.0-20220620](#)
- [emr-5.36.0-20220525](#)

Notas de la versión de Amazon EMR 5.36.0

- Se corrigieron los problemas de seguridad de `log4j2`.
- Aplicaciones compatibles: Spark 2.4.8-amzn-2, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública; el kernel de Scala no es compatible), `livy-0.7.1`, `fluentd-4.0.0`.
- Componentes compatibles - `aws-hm-client`, `emr-ddb` `aws-sagemaker-spark-sdk`, `emr-goodies`, `emr-kinesis`, `kerberos-server`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.

Clasificaciones	Descripciones
spark-defaults	Cambiar los valores en el archivo spark-defaults.conf de Spark.
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo hive-site.xml de Spark.
spark-log4j	Cambiar los valores en el archivo log4j.properties de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-5.36.0-latest

Notas de la versión: actualmente, `emr-5.36.0-latest` apunta a `emr-5.36.0-20240321`.

Regiones: `emr-5.36.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.36.0:latest`

emr-5.36.0-20240321

Notas de la versión: `emr-5.36.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-5.36.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.36.0:20240321`

emr-5.36.0-20221219

Notas de la versión: `emr-5.36.0-20221219` se lanzó el 27 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.36.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.36.0:20221219`

emr-5.36.0-20220620

Notas de la versión: `emr-5.36.0-20220620` se lanzó el 27 de julio de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.36.0-20220620` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.36.0:20220620`

emr-5.36.0-20220525

Notas de la versión: `emr-5.36.0-20220525` se lanzó el 16 de junio de 2022. Esta es la versión inicial de Amazon EMR 5.36.0.

Regiones: `emr-5.36.0-20220525` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.36.0:20220525`

Versiones de Amazon EMR en EKS 5.35.0

Las siguientes versiones de Amazon EMR 5.35.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-5.35.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-5.35.0-latest](#)
- [emr-5.35.0-20240321](#)
- [emr-5.35.0-20221219](#)
- [emr-5.35.0-20220802](#)
- [emr-5.35.0-20220307](#)

Notas de la versión de Amazon EMR 5.35.0

- Se corrigieron los problemas de seguridad de log4j2.
- Aplicaciones compatibles: Spark 2.4.8-amzn-1, Hudi 0.9.0-amzn-2, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública; el kernel de Scala no es compatible).
- Componentes compatibles - aws-hm-client (conector Glue), emr-s3-select aws-sagemaker-spark-sdk, emrfs, emr-ddb, hudi-spark.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
core-site	Cambia los valores en el archivo core-site.xml de Hadoop.
emrfs-site	Cambiar la configuración de EMRFS.
spark-metrics	Cambiar los valores en el archivo metrics.properties de Spark.
spark-defaults	Cambiar los valores en el archivo spark-defaults.conf de Spark.
spark-env	Cambiar los valores en el entorno de Spark.
spark-hive-site	Cambia los valores en el archivo hive-site.xml de Spark.
spark-log4j	Cambiar los valores en el archivo log4j.properties de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `.xml.spark-hive-site`. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-5.35.0-latest

Notas de la versión: actualmente, `emr-5.35.0-latest` apunta a `emr-5.35.0-20240321`.

Regiones: `emr-5.35.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.35.0:latest`

emr-5.35.0-20240321

Notas de la versión: `emr-5.35.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-5.35.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.35.0:20240321`

emr-5.35.0-20221219

Notas de la versión: `emr-5.35.0-20221219` se lanzó el 27 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.35.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.35.0:20221219`

emr-5.35.0-20220802

Notas de la versión: `emr-5.35.0-20220802` se lanzó el 27 de septiembre de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.35.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.35.0:20220802`

emr-5.35.0-20220307

Notas de la versión: `emr-5.35.0-20220307` se lanzó el 30 de marzo de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.35.0-20220307` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.35.0:20220307`

Versiones de Amazon EMR en EKS 5.34.0

Las siguientes versiones de Amazon EMR 5.34.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-5.34.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-5.34.0-latest](#)
- [emr-5.34.0-20240321](#)
- [emr-5.34.0-20220802](#)

Notas de la versión de Amazon EMR 5.34.0

- Aplicaciones compatibles: Spark 2.4.8-amzn-0, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública; el kernel de Scala no es compatible).

- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-5.34.0-latest

Notas de la versión: actualmente, `emr-5.34.0-latest` apunta a `emr-5.34.0-20220802`.

Regiones: `emr-5.34.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.34.0:latest`

emr-5.34.0-20240321

Notas de la versión: `emr-5.34.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-5.34.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.34.0:20240321`

emr-5.34.0-20220802

Notas de la versión: `emr-5.34.0-20220802` se lanzó el 24 de agosto de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.34.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.34.0:20220802`

emr-5.34.0-20211208

Notas de la versión: `emr-5.34.0-20211208` se lanzó el 20 de enero de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.34.0-20211208` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.34.0:20211208`

Versiones de Amazon EMR en EKS 5.33.0

Las siguientes versiones de Amazon EMR 5.33.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-5.33.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-5.33.0-latest](#)
- [emr-5.33.0-20240321](#)
- [emr-5.33.0-20221219](#)
- [emr-5.33.0-20220802](#)
- [emr-5.33.0-20211008](#)
- [emr-5.33.0-20210802](#)
- [emr-5.33.0-20210615](#)
- [emr-5.33.0-20210323](#)

Notas de la versión de Amazon EMR 5.33.0

- Nueva característica: a partir de Amazon EMR 5.33.0 en la serie 5.x, Amazon EMR en EKS es compatible con la característica de plantillas de pods de Spark. Para obtener más información, consulte [Uso de plantillas de pods](#).
- Aplicaciones compatibles: Spark 2.4.7-amzn-1, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública; el kernel de Scala no es compatible).
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.

Clasificaciones	Descripciones
spark-hive-site	Cambia los valores en el archivo hive-site.xml de Spark.
spark-log4j	Cambiar los valores en el archivo log4j.properties de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como spark-hive-site.xml. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-5.33.0-latest

Notas de la versión: actualmente, `emr-5.33.0-latest` apunta a `emr-5.33.0-20240321`.

Regiones: `emr-5.33.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:latest`

emr-5.33.0-20240321

Notas de la versión: `emr-5.33.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-5.33.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:20240321`

emr-5.33.0-20221219

Notas de la versión: `emr-5.33.0-20221219` se lanzó el 19 de enero de 2023. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-5.33.0-20221219` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:20221219`

`emr-5.33.0-20220802`

Notas de la versión: `emr-5.33.0-20220802` se lanzó el 24 de agosto de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.33.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:20220802`

`emr-5.33.0-20211008`

Notas de la versión: `emr-5.33.0-20211008` se lanzó el 9 de diciembre de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.33.0-20211008` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:20211008`

`emr-5.33.0-20210802`

Notas de la versión: `emr-5.33.0-20210802` se lanzó el 2 de agosto de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.33.0-20210802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:20210802`

emr-5.33.0-20210615

Notas de la versión: `emr-5.33.0-20210615` se lanzó el 15 de junio de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.33.0-20210615` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0:20210615`

emr-5.33.0-20210323

Notas de la versión: `emr-5.33.0-20210323` se lanzó el 23 de marzo de 2021. Esta es la versión inicial de Amazon EMR 5.33.0.

Regiones: `emr-5.33.0-20210323` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.33.0-20210323`

Versiones de Amazon EMR en EKS 5.32.0

Las siguientes versiones de Amazon EMR 5.32.0 están disponibles para Amazon EMR en EKS. Seleccione una versión específica de `emr-5.32.0-XXXX` para ver más detalles, como la etiqueta de imagen de contenedor relacionada.

- [emr-5.32.0-latest](#)
- [emr-5.32.0-20240321](#)
- [emr-5.32.0-20220802](#)
- [emr-5.32.0-20211008](#)
- [emr-5.32.0-20210802](#)
- [emr-5.32.0-20210615](#)
- [emr-5.32.0-20210129](#)
- [emr-5.32.0-20201218](#)
- [emr-5.32.0-20201201](#)

Notas de la versión de Amazon EMR 5.32.0

- Aplicaciones compatibles: Spark 2.4.7-amzn-0, Jupyter Enterprise Gateway (puntos de conexión, versión preliminar pública; el kernel de Scala no es compatible).
- Componentes compatibles: `aws-hm-client` (conector de Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Clasificaciones de configuración compatibles:

Clasificaciones	Descripciones
<code>core-site</code>	Cambia los valores en el archivo <code>core-site.xml</code> de Hadoop.
<code>emrfs-site</code>	Cambiar la configuración de EMRFS.
<code>spark-metrics</code>	Cambiar los valores en el archivo <code>metrics.properties</code> de Spark.
<code>spark-defaults</code>	Cambiar los valores en el archivo <code>spark-defaults.conf</code> de Spark.
<code>spark-env</code>	Cambiar los valores en el entorno de Spark.
<code>spark-hive-site</code>	Cambia los valores en el archivo <code>hive-site.xml</code> de Spark.
<code>spark-log4j</code>	Cambiar los valores en el archivo <code>log4j.properties</code> de Spark.

Las clasificaciones de configuración le permiten personalizar las aplicaciones. Suelen corresponder a un archivo XML de configuración de la aplicación, como `spark-hive-site.xml`. Para obtener más información, consulte [Configuración de aplicaciones](#).

emr-5.32.0-latest

Notas de la versión: actualmente, `emr-5.32.0-latest` apunta a `emr-5.32.0-20240321`.

Regiones: `emr-5.32.0-latest` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.32.0:latest`

`emr-5.32.0-20240321`

Notas de la versión: `emr-5.32.0-20240321` se lanzó el 11 de marzo de 2024. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados y se han hecho correcciones críticas.

Regiones: `emr-5.32.0-20240321` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.32.0:20240321`

`emr-5.32.0-20220802`

Notas de la versión: `emr-5.32.0-20220802` se lanzó el 24 de agosto de 2022. En comparación con la versión anterior, esta versión se ha actualizado con los paquetes de Amazon Linux recientemente actualizados.

Regiones: `emr-5.32.0-20220802` está disponible en todas las regiones compatibles con Amazon EMR en EKS. Para obtener más información, consulte [Puntos de conexión de servicio de Amazon EMR en EKS](#).

Etiqueta de imagen de contenedor: `emr-5.32.0:20220802`

`emr-5.32.0-20211008`

Notas de la versión: `emr-5.32.0-20211008` se lanzó el 9 de diciembre de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.32.0-20211008` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-5.32.0:20211008`

emr-5.32.0-20210802

Notas de la versión: `emr-5.32.0-20210802` se lanzó el 2 de agosto de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.32.0-20210802` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-5.32.0:20210802`

emr-5.32.0-20210615

Notas de la versión: `emr-5.32.0-20210615` se lanzó el 15 de junio de 2021. En comparación con la versión anterior, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.32.0-20210615` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-5.32.0:20210615`

emr-5.32.0-20210129

Notas de la versión: `emr-5.32.0-20210129` se lanzó el 29 de enero de 2021. En comparación con `emr-5.32.0-20201218`, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.32.0-20210129` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-5.32.0-20210129`

emr-5.32.0-20201218

Notas de la versión: `5.32.0-20201218` se lanzó el 18 de diciembre de 2020. En comparación con `5.32.0-20201201`, esta versión contiene correcciones de problemas y actualizaciones de seguridad.

Regiones: `emr-5.32.0-20201218` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-5.32.0-20201218`

`emr-5.32.0-20201201`

Notas de la versión: `5.32.0-20201201` se lanzó el 1 de diciembre de 2020. Esta es la versión inicial de Amazon EMR 5.32.0.

Regiones: `5.32.0-20201201` está disponible en las siguientes regiones: Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia-Pacífico (Tokio), Europa (Irlanda) y América del Sur (São Paulo).

Etiqueta de imagen de contenedor: `emr-5.32.0-20201201`

Historial del documento

En la siguiente tabla, se describen los cambios importantes que se han hecho en la documentación desde la última versión de Amazon EMR en EKS. Para obtener más información sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS.

Cambio	Descripción	Fecha
Actualizar contenido	Políticas administradas para Amazon EMR en EKS — Permisos adicionales para AmazonEMRContainersServiceRolePolicy .	3 de febrero de 2025
Nueva versión	Amazon EMR en las versiones 7.6.0 de EKS	10 de enero de 2025
Nueva versión	Amazon EMR en las versiones 7.5.0 de EKS	21 de noviembre de 2024
Nueva versión	Amazon EMR en las versiones 7.4.0 de EKS	13 de noviembre de 2024
Nueva versión	Versiones 7.3.0 de Amazon EMR en EKS	16 de octubre de 2024
Nueva versión	Versiones 7.2.0 de Amazon EMR en EKS	25 de julio de 2024
Nueva versión	Versiones 7.1.0 de Amazon EMR en EKS	17 de abril de 2024
Nueva versión	Versiones 7.0.0 de Amazon EMR en EKS	22 de diciembre de 2023
Nueva versión	Versiones 6.15.0 de Amazon EMR en EKS	17 de noviembre de 2023

Cambio	Descripción	Fecha
Nueva versión	Versiones de Amazon EMR en EKS 6.14.0	17 de octubre de 2023
Actualizar contenido	Cambie el nombre de los “puntos de conexión administrados” por puntos de conexión interactivos . Disponibilidad general de los puntos de conexión interactivos	29 de septiembre de 2023
Nueva versión	Versiones de Amazon EMR en EKS 6.13.0 , y documentos de vista previa pública de Ejecución de trabajos de Flink con Amazon EMR en EKS	12 de septiembre de 2023
Nueva versión	Versiones de Amazon EMR en EKS 6.12.0	21 de julio de 2023
Contenido nuevo	Se ha agregado Uso de Volcano como programador personalizado para Apache Spark en Amazon EMR en EKS	13 de junio de 2023
Contenido nuevo	Se ha agregado Uso de Volcano como programador personalizado para Apache Spark en Amazon EMR en EKS	13 de junio de 2023
Contenido nuevo	Se ha agregado Uso de la rotación de los registros de contenedores de Spark	12 de junio de 2023
Actualizar contenido	Se actualizó la documentación sobre imágenes personalizadas para encontrar información sobre imágenes base en Amazon ECR Public Gallery.	8 de junio de 2023
Nueva versión	Versiones de Amazon EMR en EKS 6.11.0	8 de junio de 2023

Cambio	Descripción	Fecha
Contenido nuevo	Se agregó Ejecución de trabajos de Spark con el operador de Spark y se reorganizaron las secciones de ejecuciones de trabajos en Ejecución de trabajos de Spark con Amazon EMR en EKS .	5 de junio de 2023
Contenido nuevo	Se han agregado dos secciones: Uso del escalado automático vertical con trabajos de Spark de Amazon EMR y Uso de cuadernos Jupyter autoalojados	4 de mayo de 2023
Página de historial de documentos	Se creó una página de historial de documentos para Amazon EMR en EKS.	13 de marzo de 2023
Página de políticas administradas	Creó una página de políticas administradas para Amazon EMR en EKS.	13 de marzo de 2023

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.