



Guía del desarrollador de Managed Service para Apache Flink

Managed Service para Apache Flink



Managed Service para Apache Flink: Guía del desarrollador de Managed Service para Apache Flink

Copyright © 2024 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 comerciales que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, relacionados o patrocinados por Amazon.

Table of Contents

.....	xvi
¿Qué es Managed Service para Apache Flink?	1
Elegir un servicio gestionado para Apache Flink o un servicio gestionado para Apache Flink Studio	1
Elegir qué API de Apache Flink usar en Managed Service for Apache Flink	3
Elegir una API de Flink	3
Introducción	4
Cómo funciona	6
Programar la aplicación Apache Flink	6
API de Datastream	6
API de tabla	7
Creación de su aplicación de Managed Service para Apache Flink	8
Creación de aplicaciones	8
Creación del código de su aplicación de Managed Service para Apache Flink	8
Creación de su aplicación de Managed Service para Apache Flink	10
Iniciar su aplicación de Managed Service para Apache Flink	11
Verificación de su aplicación Managed Service para Apache Flink	11
Ejecución de aplicaciones	12
Solicitud y estado del trabajo	12
Cargas de trabajo en lotes	14
Recursos de aplicaciones	14
Recursos de aplicaciones de Managed Service para Apache Flink	14
Recursos de la aplicación de Apache Flink	15
API de DataStream	16
DataStream Conectores API	17
Operadores de la API de DataStream	31
Marcas de tiempo de la API de DataStream	33
API de tabla	33
Conectores de la API de tabla	34
Atributos de tiempo de la API de tabla	35
Uso de Python	36
Programar una aplicación	37
Creación de una aplicación	40
Supervisión	41

Propiedades de tiempo de ejecución	42
Trabajar con las propiedades de tiempo de ejecución en la CLI	43
Trabajar con las propiedades de tiempo de ejecución en la consola	43
Acceder a propiedades de tiempo de ejecución en un servicio gestionado para la aplicación Apache Flink	46
Tolerancia a errores	47
Configuración de puntos de control	48
Ejemplos de la API de puntos de control	49
Instantáneas	52
Escalado	57
Configuración del paralelismo de aplicaciones y la KPU ParallelismPer	58
Asignación de unidades de procesamiento de Kinesis	58
Actualización del paralelismo de la aplicación	59
Escalado automático	61
Etiquetado	63
Adición de etiquetas al crear una aplicación	64
Adición o actualización de etiquetas para una aplicación existente	64
Mostrar las etiquetas de una aplicación	65
Eliminación de etiquetas de una aplicación	65
Uso de CloudFormation con Managed Service para Apache Flink	65
Antes de empezar	65
Escritura de una función de Lambda	66
Creación de rol de Lambda	68
Invocación de la función de Lambda	68
Invocación de la función de Lambda	69
Panel de Apache Flink	75
Acceder al panel de control de Apache Flink de su aplicación	76
Versiones de lanzamiento	78
Amazon Managed Service para Apache Flink versión 1.15.2	78
Cambios en Amazon Managed Service para Apache Flink con Apache Flink 1.15	80
Componentes	80
Bloc de notas de Studio	82
Crear un bloc de notas de Studio	83
Análisis interactivo de los datos de streaming	84
Intérpretes de Flink	85
Variables de entorno de la tabla de Apache Flink	86

Implementación como una aplicación con un estado duradero	86
Criterios de Scala/Python	88
Criterios SQL	88
Permisos de IAM	89
Conectores y dependencias	89
Conectores por defecto	89
Dependencias y conectores personalizados	91
Funciones definidas por el usuario	92
Consideraciones con funciones definidas por el usuario	93
Activación de puntos de control	95
Establecer el intervalo de puntos de control	95
Establecer el tipo de punto de control	95
Trabajar con AWS Glue	96
Propiedades de la tabla	96
Ejemplos y tutoriales	98
Tutorial sobre cómo crear un bloc de notas de Studio	99
Implementación como una aplicación con estado duradero	119
Ejemplos	123
Solución de problemas	135
Detener una aplicación atascada	135
Implementación como una aplicación con un estado duradero en una VPC sin acceso a Internet	135
Reducción del eploy-as-app tamaño D y del tiempo de construcción	136
Cancelación de trabajos	138
Reiniciar el intérprete Apache Flink	139
Apéndice: Creación de políticas de IAM personalizadas	139
AWS Glue	140
CloudWatch Registros	141
Flujos de Kinesis	142
Clústeres de Amazon MSK	144
Cómo empezar (DataStream API)	145
Componentes de la aplicación	145
Requisitos previos	146
Paso 1: Configurar una cuenta	146
Registrarse en una Cuenta de AWS	146
Crear un usuario administrativo	147

Conceder acceso programático	148
Paso siguiente	150
Paso 2: Configuración de la AWS CLI	150
Paso siguiente	152
Paso 3: Crear una aplicación	152
Crear dos Amazon Kinesis Data Streams	153
Escribir registros de muestra en el flujo de entrada	153
Descargar y examinar el código de Java de streaming de Apache Flink	154
Compilar el código de la aplicación	155
Cargar el código de Java de streaming de Apache Flink	156
Crear y ejecutar la aplicación de Managed Service para Apache Flink	157
Paso siguiente	169
Paso 4: Eliminación	169
Elimine su aplicación de Managed Service para Apache Flink	170
Eliminar sus flujos de datos de Kinesis	170
Eliminar el objeto y el bucket de Amazon S3	170
Eliminar sus recursos de IAM	170
CloudWatch Elimine sus recursos	171
Paso siguiente	171
Paso 5: Sigüientes pasos	171
Introducción (API de tablas)	173
Componentes de la aplicación	173
Requisitos previos	174
Crear una aplicación	174
Crear recursos dependientes	175
Escribir registros de muestra en el flujo de entrada	176
Descargar y examinar el código de Java de streaming de Apache Flink	177
Compilar el código de la aplicación	179
Cargar el código de Java de streaming de Apache Flink	180
Crear y ejecutar la aplicación de Managed Service para Apache Flink	180
Paso siguiente	185
Eliminación	185
Elimine su aplicación de Managed Service para Apache Flink	185
Elimine su clúster de Amazon MSK	186
Eliminar su VPC	186
Eliminar objetos y el bucket de Amazon S3	186

Eliminar sus recursos de IAM	186
CloudWatch Elimine sus recursos	187
Paso siguiente	187
Sigüientes pasos	187
Introducción (Python)	188
Introducción a Pyflink: el intérprete de Python para Apache Amazon Web Services	188
Componentes de la aplicación	189
Requisitos previos	189
Crear una aplicación	190
Crear recursos dependientes	190
Escribir registros de muestra en el flujo de entrada	192
Crear y examinar el código de Python de streaming de Apache Flink	193
Añadir dependencias de terceros a las aplicaciones de Python	195
Cargar el código de Python de streaming de Apache Flink	196
Crear y ejecutar la aplicación de Managed Service para Apache Flink	198
Paso siguiente	202
Eliminación	202
Elimine su aplicación de Managed Service para Apache Flink	203
Eliminar sus flujos de datos de Kinesis	203
Eliminar objetos y el bucket de Amazon S3	203
Eliminar sus recursos de IAM	204
CloudWatch Elimine sus recursos	204
Introducción (Scala)	205
Crear recursos dependientes	205
Escribir registros de muestra en la secuencia de entrada	207
Descargue y examine el código de la aplicación	208
Compilar y cargar el código de la aplicación	209
Crear y ejecutar la aplicación (consola)	211
Crear la aplicación de	211
Configurar la aplicación	211
Editar la política de IAM	213
Ejecutar la aplicación	215
Detener la aplicación	215
Compilar y ejecutar la aplicación (CLI)	215
Crear una política de permisos	215
Crear una política de IAM	217

Crear la aplicación	219
Iniciar la aplicación	220
Detener la aplicación	221
Añadir una opción de registro de CloudWatch	221
Actualizar propiedades de entorno	221
Actualizar el código de la aplicación	222
Eliminación	223
Elimine su aplicación de Managed Service para Apache Flink	224
Eliminar sus flujos de datos de Kinesis	224
Eliminar el objeto y el bucket de Amazon S3	224
Eliminar sus recursos de IAM	224
Eliminar sus recursos de CloudWatch	225
Uso de Apache Beam	226
Uso de Apache Beam con Managed Service para Apache Flink	226
Capacidades de Beam	226
Crear una aplicación con Apache Beam	227
Crear recursos dependientes	227
Escribir registros de muestra en la secuencia de entrada	228
Descargue y examine el código de la aplicación	229
Compilar el código de la aplicación	230
Cargar el código de Java de streaming de Apache Flink	231
Crear y ejecutar la aplicación de Managed Service para Apache Flink	231
Eliminación	235
Pasos siguientes	237
Talleres de formación, laboratorios e implementaciones de soluciones	238
Desarrollo de aplicaciones de Apache Flink localmente antes de implementarlas en Managed Service para Apache Flink	238
Detección de eventos con Managed Service para Apache Flink Studio	238
Solución de datos de streaming de AWS	239
Laboratorio Clickstream	239
Escalado personalizado	239
CloudWatch Panel de control	240
Amazon MSK	240
Más servicios gestionados para Apache Flink Solutions en GitHub	240
Utilidades	241
Gestor de instantáneas	241

Evaluación comparativa	241
Ejemplos	242
DataStream Ejemplos de API	242
Ventana de saltos de tamaño constante	243
Ventana deslizante	252
Receptor S3	262
Replicación MSK	277
Consumidor de EFO	283
Receptor de Kinesis Data Firehose	294
entre cuentas	310
Truststore personalizado	319
Ejemplos de Python	329
Ventana de saltos de tamaño constante	329
Ventana deslizante	340
Receptor S3	350
Ejemplo de Scala	362
Ventana de saltos de tamaño constante	362
Ventana deslizante	379
Receptor S3	397
Seguridad	415
Protección de los datos	416
Cifrado de datos	416
Identidad y gestión de acceso	417
Público	417
Autenticación con identidades	418
Administración de acceso mediante políticas	422
Cómo funciona Amazon Managed Service para Apache Flink con IAM	425
Ejemplos de políticas basadas en identidades	433
Solución de problemas	436
Prevención de la sustitución confusa entre servicios	438
Supervisión	440
Validación de la conformidad	440
FedRAMP	441
Resiliencia	441
Recuperación de desastres	441
Control de versiones	442

Seguridad de infraestructuras	443
Prácticas recomendadas de seguridad	443
Implementación del acceso a los privilegios mínimos	443
Uso de los roles de IAM para obtener acceso a otros servicios de Amazon	444
Implementación del cifrado en el servidor en recursos dependientes	444
Se utiliza para supervisar las llamadas CloudTrail a la API	444
Registro y supervisión	445
Registro	446
Consulta de registros con Logs Insights CloudWatch	446
Monitorización	446
Configuración del registro	448
Configuración del CloudWatch registro mediante la consola	449
Configuración del CloudWatch registro mediante la CLI	449
Niveles de monitoreo de aplicaciones	454
Prácticas recomendadas de registro	455
Solución de problemas de registro	456
Paso siguiente	456
Análisis de registros	456
Ejecutar una consulta de muestra	457
Consultas de ejemplo	458
Métricas y dimensiones en Managed Service para Apache Flink	461
Métricas de aplicación	461
Métricas del conector de flujo de datos de Kinesis	491
Métricas de Amazon MSK Connector	492
Métricas de Apache Zeppelin	494
Visualización de métricas CloudWatch	495
Métricas	496
Métricas personalizadas	498
Alarmas	502
Escribir mensajes personalizados	514
Escriba en los CloudWatch registros mediante Log4J	514
Escriba en los CloudWatch registros con SLF4J	515
Uso de AWS CloudTrail	516
Servicio gestionado para Apache Flink Information en CloudTrail	516
Comprender las entradas de archivos de registro de Managed Service para Apache Flink ..	518
Desempeño	520

Solución de problemas de desempeño	520
La ruta de datos	520
Solución de problemas de desempeño	521
Prácticas recomendadas de desempeño	523
Administrar correctamente el escalado	523
Supervisar el uso de los recursos de dependencia externa	526
Ejecute la aplicación Apache Flink localmente	526
Monitoreo del desempeño	526
Monitoreo de desempeño con métricas de CloudWatch	526
Supervisión del desempeño mediante registros CloudWatch y Alarmas Alarms	526
Cuota	528
Mantenimiento	530
Establezca un UUID para todos los operadores	532
Preparación para la producción	533
Aplicaciones de pruebas de carga	533
Máximo paralelismo	534
Establezca un UUID para todos los operadores	534
Prácticas recomendadas	535
Tolerancia a fallos: puntos de control y puntos de almacenamiento	535
Versiones de conector no compatibles	536
Rendimiento y paralelismo	536
Establecer el paralelismo por operador	537
Registro	538
Codificación	538
Administración de credenciales	539
Lectura de fuentes con pocas particiones	539
Intervalo de actualización del bloc de notas de Studio	540
Rendimiento óptimo del bloc de notas de Studio	540
Cómo afectan las estrategias de marcas de agua y las particiones inactivas a las ventanas temporales	540
Resumen	542
Ejemplo	542
Establezca un UUID para todos los operadores	551
Añádalo ServiceResourceTransformer al complemento Maven Shade	552
Stateful Functions de Apache Flink	554
Plantilla de aplicación de Apache Flink	554

Ubicación de la configuración del módulo	555
Versiones anteriores	556
Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink	557
Creación de aplicaciones con Apache Flink 1.8.2	558
Creación de aplicaciones con Apache Flink 1.6.2	559
Actualización de aplicaciones	560
Conectores disponibles en Apache Flink 1.6.2 y 1.8.2	560
Introducción: Flink 1.13.2	561
Componentes de la aplicación	561
Requisitos previos	562
Paso 1: Configurar una cuenta	562
Paso siguiente	566
Paso 2: Configuración de la AWS CLI	566
Paso 3: Crear una aplicación	568
Paso 4: Eliminación	585
Paso 5: Sigüientes pasos	587
Introducción: Flink 1.11.1	588
Componentes de la aplicación	588
Requisitos previos	589
Paso 1: Configurar una cuenta	590
Paso 2: Configuración de la AWS CLI	593
Paso 3: Crear una aplicación	595
Paso 4: Eliminación	612
Paso 5: Sigüientes pasos	614
Introducción: Flink 1.8.2	615
Componentes de la aplicación	145
Requisitos previos	616
Paso 1: Configurar una cuenta	617
Paso 2: Configuración de la AWS CLI	620
Paso 3: Crear una aplicación	622
Paso 4: Eliminación	639
Introducción: Flink 1.6.2	641
Componentes de la aplicación	641
Requisitos previos	642
Paso 1: Configurar una cuenta	643
Paso 2: Configuración de la AWS CLI	646

Paso 3: Crear una aplicación	648
Paso 4: Eliminación	665
Configuración de Flink	667
Configuración de Apache Flink	667
Backend de estado	668
Creación de puntos de control	668
Usar punto de guardado	670
Tamaños de montón	670
Deblotear el búfer	670
Propiedades de configuración de Flink modificables	671
Tolerancia a errores	671
Puntos de control y backends de estado	671
Creación de puntos de control	671
Métricas nativas de RocksDB	671
Opciones avanzadas de backends de estado	673
Opciones completas de TaskManager	673
Configuración de memoria	673
RPC/Akka	674
Cliente	674
Opciones avanzadas de clústeres	674
Configuraciones del sistema de archivos	674
Opciones avanzadas de tolerancia a errores	675
Configuración de memoria	673
Métricas	675
Opciones avanzadas para el cliente y el punto de conexión REST	675
Opciones de seguridad SSL avanzadas	675
Opciones avanzadas de programación	675
Opciones avanzadas para la interfaz de usuario web de Flink	675
Visualización de las propiedades configuradas de Flink	675
Uso de una Amazon VPC	677
Conceptos de Amazon VPC	677
Permisos de aplicación de la VPC	678
Política de permisos para acceder a una Amazon VPC	679
Acceso a Internet y a servicios	680
Información relacionada	681
API de VPC	681

CreateApplication	681
AddApplicationVpcConfiguration	682
DeleteApplicationVpcConfiguration	683
UpdateApplication	683
Ejemplo: uso de una VPC	684
Solución de problemas	685
Solución de problemas de desarrollo	685
Gráficos de Apache Flink Flame	685
Problema del proveedor de credenciales con el conector EFO 1.15.2	686
Aplicaciones con conectores Kinesis no compatibles	686
Error de compilación: "No se pudieron resolver las dependencias del proyecto"	689
Opción no válida: "kinesisanalyticsv2"	689
UpdateApplication La acción no es volver a cargar el código de la aplicación	689
S3 StreamingFileSink FileNotFoundException	690
FlinkKafkaConsumer problema con stop with savepoint	692
Bloqueo de Flink 1.15 Async Sink	692
El procesamiento del origen de Amazon Kinesis Data Streams no funciona durante la partición nueva.	702
Solución de problemas de tiempo de ejecución	703
Herramientas para solucionar problemas	703
Problemas con las aplicaciones	703
La aplicación se está reiniciando	708
El rendimiento es demasiado lento	711
Crecimiento de estado ilimitado	712
Operadores vinculados a E/S	714
Limitación ascendente o de origen desde un flujo de datos de Kinesis	714
Puntos de control	715
Se está agotando el tiempo para llegar al punto de control	722
Fallo en el punto de control (Beam)	723
Resistencia	725
Sesgo de datos	727
Sesgo de estado	728
Integración con recursos en diferentes regiones	728
Historial de documentos	729
Ejemplo de código de API	736
AddApplicationCloudWatchLoggingOption	737

AddApplicationInput	737
AddApplicationInputProcessingConfiguration	738
AddApplicationOutput	739
AddApplicationReferenceDataSource	739
AddApplicationVpcConfiguration	740
CreateApplication	741
CreateApplicationSnapshot	742
DeleteApplication	742
DeleteApplicationCloudWatchLoggingOption	742
DeleteApplicationInputProcessingConfiguration	743
DeleteApplicationOutput	743
DeleteApplicationReferenceDataSource	743
DeleteApplicationSnapshot	744
DeleteApplicationVpcConfiguration	744
DescribeApplication	744
DescribeApplicationSnapshot	744
DiscoverInputSchema	745
ListApplications	745
ListApplicationSnapshots	746
StartApplication	746
StopApplication	746
UpdateApplication	747
Referencia de la API	748

Amazon Managed Service para Apache Flink Amazon se denominaba anteriormente Amazon Kinesis Data Analytics para Apache Flink.

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.

¿Qué es Amazon Managed Service para Apache Flink?

Con Amazon Managed Service para Apache Flink, puede usar Java, Scala, Python o SQL para procesar y analizar los datos de streaming. El servicio le permite crear y ejecutar código en fuentes de streaming y fuentes estáticas para realizar análisis de series temporales, alimentar cuadros de mando en tiempo real y métricas.

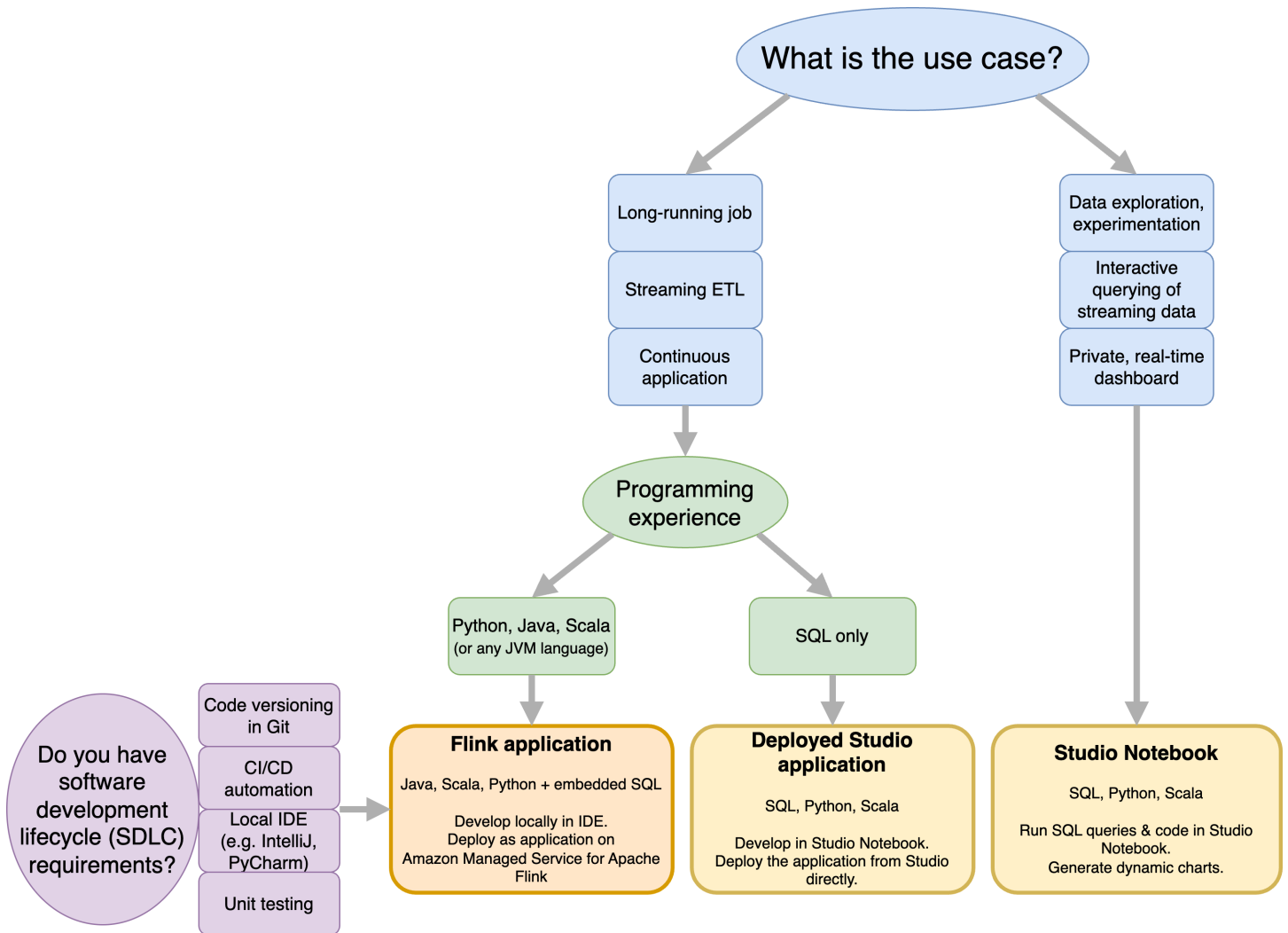
[Puede crear aplicaciones con el lenguaje que prefiera en Managed Service for Apache Flink mediante bibliotecas de código abierto basadas en Apache Flink.](#) Apache Flink es un marco y un motor usados habitualmente para procesar flujos de datos.

Managed Service para Apache Flink proporciona la infraestructura subyacente para sus aplicaciones de Apache Flink. Gestiona las capacidades principales, como el aprovisionamiento de recursos informáticos, la resiliencia de la conmutación por error de AZ, la computación paralela, el escalado automático y las copias de seguridad de las aplicaciones (implementadas como puntos de control e instantáneas). Puede utilizar las características de programación de alto nivel de Flink (como operadores, funciones, fuentes y receptores) del mismo modo que las utiliza cuando aloja usted mismo la infraestructura de Flink.

Elegir un servicio gestionado para Apache Flink o un servicio gestionado para Apache Flink Studio

Tiene dos opciones para ejecutar sus trabajos de Flink con Amazon Managed Service para Apache Flink. Con [Managed Service for Apache Flink](#), puede crear aplicaciones de Flink en Java, Scala o Python (y SQL integrado) mediante un IDE de su elección y las API Apache Flink Datastream o Table. Con [Managed Service for Apache Flink Studio](#), puede consultar flujos de datos de forma interactiva en tiempo real y crear y ejecutar fácilmente aplicaciones de procesamiento de flujos utilizando SQL, Python y Scala estándares.

Puede seleccionar el método que mejor se adapte a su caso de uso. Si no está seguro, esta sección le ofrecerá una guía de alto nivel para ayudarle.



Antes de decidir si usar Amazon Managed Service para Apache Flink o Amazon Managed Service para Apache Flink Studio, debes considerar tu caso de uso.

Si planea utilizar una aplicación de larga duración que se encargue de cargas de trabajo como la transmisión de ETL o las aplicaciones continuas, debería considerar la posibilidad de utilizar el [servicio gestionado para Apache Flink](#). Esto se debe a que puede crear su aplicación Flink utilizando las API de Flink directamente en el IDE que elija. El desarrollo local con su IDE también garantiza que pueda aprovechar los procesos y herramientas comunes del ciclo de vida de desarrollo de software (SDLC), como el control de versiones de código en Git, la automatización de CI/CD o las pruebas unitarias.

Si está interesado en la exploración de datos ad hoc, desea consultar datos de streaming de forma interactiva o crear paneles privados en tiempo real, [Managed Service for Apache Flink Studio](#) le ayudará a cumplir estos objetivos con tan solo unos clics. Los usuarios familiarizados con SQL pueden plantearse implementar directamente desde Studio una aplicación de larga duración.

Note

Puede convertir su portátil Studio en una aplicación de larga duración. Sin embargo, si desea integrarlas con sus herramientas de SDLC, como el control de versiones de código en Git y la automatización de CI/CD, o técnicas como las pruebas unitarias, le recomendamos Managed Service for Apache Flink que utilice el IDE que prefiera.

Elegir qué API de Apache Flink usar en Managed Service for Apache Flink

Puede crear aplicaciones con Java, Python y Scala en Managed Service for Apache Flink mediante las API de Apache Flink en el IDE que elija. [Puede encontrar instrucciones sobre cómo crear aplicaciones mediante la API Flink Datastream y Table en la documentación.](#) Puede seleccionar el idioma en el que crea su aplicación Flink y las API que utiliza para satisfacer mejor las necesidades de su aplicación y sus operaciones. Si no está seguro, esta sección proporciona una guía de alto nivel para ayudarle.

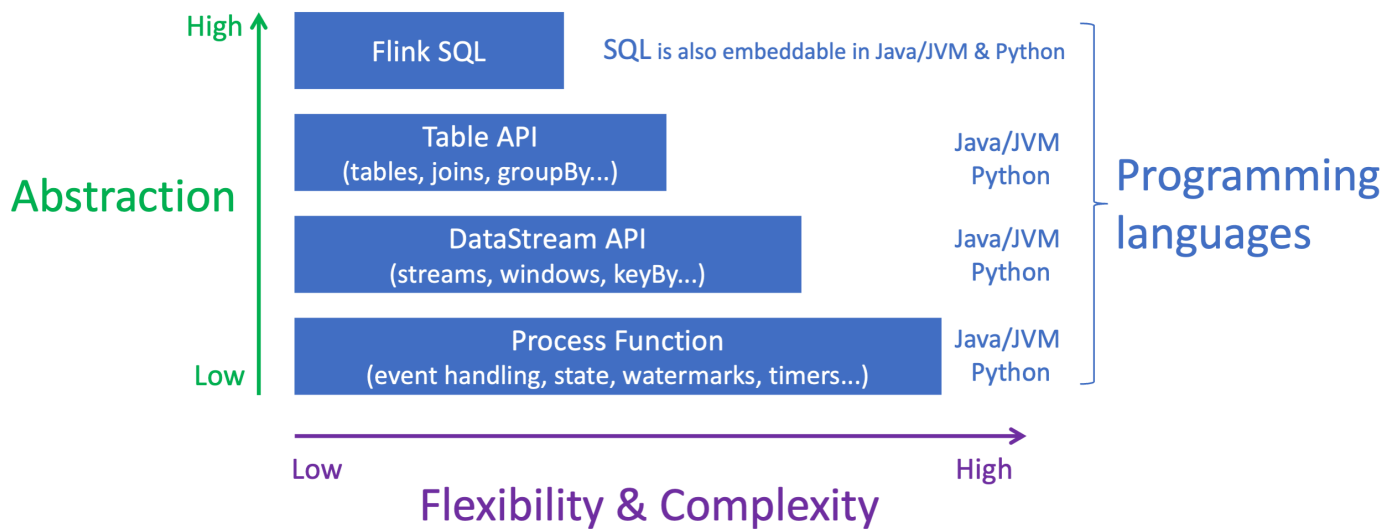
Elegir una API de Flink

Las API de Apache Flink tienen diferentes niveles de abstracción que pueden afectar a la forma en que decida crear su aplicación. Son expresivas y flexibles y se pueden usar juntas para crear su aplicación. No tiene que usar solo una API de Flink. Puede obtener más información sobre las API de Flink en la documentación de [Apache Flink](#).

Flink ofrece cuatro niveles de abstracción de API: Flink SQL, Table API, DataStream API y Process Function, que se utiliza junto con la API. DataStream Todos ellos son compatibles con Amazon Managed Service para Apache Flink. Se recomienda empezar con un nivel de abstracción más alto siempre que sea posible; sin embargo, algunas funciones de Flink solo están disponibles con la [API de Datastream](#), donde puede crear su aplicación en Java, Python o Scala. Deberías considerar usar la API de Datastream si:

- Necesita un control detallado sobre el estado
- Desea aprovechar la posibilidad de llamar a una base de datos externa o a un punto final de forma asíncrona (por ejemplo, para realizar inferencias)
- Desea utilizar temporizadores personalizados

Apache Flink APIs



Note

Elegir un idioma con la API de Datastream:

- SQL se puede incrustar en cualquier aplicación de Flink, independientemente del lenguaje de programación elegido.
- Si planea usar la DataStream API, Python no admite todos los conectores.
- Si necesita baja latencia y alto rendimiento, debería considerar Java/Scala independientemente de la API.
- Si planea usar Async IO en la API de Process Functions, necesitará usar Java.

Introducción

Puede empezar por crear una aplicación de Managed Service para Apache Flink que lea y procese continuamente datos de streaming. A continuación, cree el código con el IDE que prefiera y pruébelo con datos de streaming en directo. También puede configurar los destinos a los que desea que Managed Service para Apache Flink envíe los resultados.

Para comenzar, le recomendamos que lea las secciones siguientes:

- [Managed Service para Apache Flink: cómo funciona](#)

- [Introducción a Amazon Managed Service para Apache Flink \(DataStream API\)](#)

Como alternativa, puede empezar por crear una libreta de servicios gestionados para Apache Flink Studio que le permita consultar flujos de datos de forma interactiva en tiempo real y crear y ejecutar fácilmente aplicaciones de procesamiento de flujos utilizando SQL, Python y Scala estándares. Con unos pocos clics en la AWS Management Console, puede lanzar un bloc de notas sin servidor para consultar flujos de datos y obtener resultados en cuestión de segundos. Para comenzar, le recomendamos que lea las secciones siguientes:

- [Uso de un bloc de notas de Studio con Managed Service para Apache Flink](#)
- [Crear un bloc de notas de Studio](#)

Managed Service para Apache Flink: cómo funciona

Managed Service para Apache Flink es un servicio de Amazon completamente gestionado, que permite usar aplicaciones de Apache Flink para procesar datos de streaming.

Programar la aplicación Apache Flink

Una aplicación Apache Flink es una aplicación Java o Scala que se crea con el marco Apache Flink. La aplicación Apache Flink se crea y compila de forma local.

Las aplicaciones utilizan principalmente la [API DataStream](#) o la [Tabla API](#). Las demás API de Apache Flink también están disponibles para su uso, pero se utilizan con menos frecuencia para crear aplicaciones de streaming.

Las características de las dos API son las siguientes:

API de Datastream

El modelo de programación de la API de DataStream de Apache Flink se basa en dos componentes:

- Flujo de datos: la representación estructurada de un flujo continuo de registros de datos.
- Operador de transformación: toma uno o más flujos de datos como entrada y produce uno o más flujos de datos como salida.

Las aplicaciones creadas con la API DataStream hacen lo siguiente:

- Leen los datos de un origen de datos (como un flujo de Kinesis o un tema de Amazon MSK).
- Aplican transformaciones a los datos, como el filtrado, la agregación o el enriquecimiento.
- Escriben los datos transformados en un receptor de datos.

Las aplicaciones que utilizan la API DataStream se pueden escribir en Java o Scala y pueden leer desde un flujo de datos de Kinesis, un tema de Amazon MSK o un origen personalizado.

La aplicación procesa los datos mediante un conector. Apache Flink utiliza los siguientes tipos de conectores:

- Origen: conector que se utiliza para leer datos externos.
- Receptor: conector que se utiliza para escribir en ubicaciones externas.
- Operador: conector que se utiliza para procesar datos dentro de la aplicación.

Una aplicación típica consta de al menos un flujo de datos con un origen, un flujo de datos con uno o más operadores y al menos un receptor de datos.

Para obtener más información sobre el uso de la API de DataStream, consulte [API de DataStream](#).

API de tabla

El modelo de programación de la API Apache Flink se basa en los siguientes componentes:

- Entorno de tablas: una interfaz para los datos subyacentes que se utiliza para crear y alojar una o más tablas.
- Tabla: objeto que proporciona acceso a una tabla o vista de SQL.
- Origen de tabla: se utiliza para leer datos de una fuente externa, como un tema de Amazon MSK.
- Función de tabla: consulta SQL o llamada a la API que se utiliza para transformar datos.
- Receptor de tabla: se utiliza para escribir datos en una ubicación externa, como un bucket de Amazon S3.

Las aplicaciones creadas con la API de tabla hacen lo siguiente:

- Crean un `TableEnvironment` conectándose a un `Table Source`.
- Crean una tabla en el `TableEnvironment` mediante consultas SQL o funciones de la API de tablas.
- Ejecutan una consulta en la tabla mediante la API de tabla o SQL.
- Aplican transformaciones a los resultados de la consulta mediante funciones de tabla o consultas SQL.
- Escriben los resultados de la consulta o función en un `Table Sink`.

Las aplicaciones que utilizan la API de tablas se pueden escribir en Java o Scala, y pueden consultar datos mediante llamadas a la API o consultas SQL.

Para obtener más información sobre el uso de la API de tabla, consulte [API de tabla](#).

Creación de su aplicación de Managed Service para Apache Flink

Managed Service para Apache Flink es un servicio de AWS que crea un entorno para brindar hosting a su aplicación Apache Flink y lo proporciona con las siguientes configuraciones:

- [Propiedades de tiempo de ejecución](#): Parámetros que puede proporcionar a su aplicación. Puede cambiar estos parámetros sin tener que volver a compilar el código de la aplicación.
- [Tolerancia a errores](#): cómo se recupera la aplicación de las interrupciones y se reinicia.
- [Registro y supervisión](#): cómo registra su aplicación los eventos en CloudWatch Logs.
- [Escalado](#): cómo aprovisiona su aplicación los recursos informáticos.

Creación de una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI. Para comenzar a crear una aplicación de Managed Service para Apache Flink, consulte [Cómo empezar \(DataStream API\)](#).

Creación de una aplicación de Managed Service para Apache Flink

En este tema, se incluye información acerca de cómo crear una aplicación de Managed Service para Apache Flink.

Este tema contiene las siguientes secciones:

- [Creación del código de su aplicación de Managed Service para Apache Flink](#)
- [Creación de su aplicación de Managed Service para Apache Flink](#)
- [Iniciar su aplicación de Managed Service para Apache Flink](#)
- [Verificación de su aplicación Managed Service para Apache Flink](#)


Creación del código de su aplicación de Managed Service para Apache Flink

En esta sección se describen los componentes que debe utilizar para crear el código de su aplicación Managed Service para Apache Flink.

Le recomendamos que utilice la última versión compatible de Apache Flink para el código de la aplicación. La última versión de Apache Flink compatible con Managed Service para Apache Flink

es la 1.15.2. Para obtener información sobre la actualización de aplicaciones Managed Service para Apache Flink, consulte [Actualización de aplicaciones](#).

El código de la aplicación se debe crear con [Apache Maven](#). Un proyecto de Apache Maven utiliza un archivo `pom.xml` para especificar las versiones de los componentes que utiliza.

 Note

Managed Service para Apache Flink admite archivos JAR de hasta 512 MB de tamaño. Si utiliza un archivo JAR de un tamaño superior a este, la aplicación no podrá iniciarse.

Utilice las siguientes versiones de los componentes para las aplicaciones de Managed Service para Apache Flink:

Componente	Versión
Java	11 (recomendado)
Scala	Consulte la nota de desacoplamiento de Scala a continuación
Servicio gestionado para Apache Flink Runtime (<code>aws-kinesisanalytics-runtime</code>)	1.2.0
AWSConnector Kinesis () <code>flink-connector-kinesis</code>	1.15.2
Apache Beam (solo aplicaciones Beam)	2.33.0, con Jackson versión 2.12.2

A partir de la versión 1.15, Flink no incluye Scala. Las aplicaciones ahora pueden usar la API de Java desde cualquier versión de Scala. Deberá incluir la biblioteca estándar de Scala que elija en sus aplicaciones de Scala.

Para ver un ejemplo de un archivo `pom.xml` de una aplicación de Managed Service para Apache Flink que utiliza la versión 1.15.2 de Apache Flink, consulte la [aplicación de introducción de Managed Service para Apache Flink](#).

Para obtener información sobre cómo crear una aplicación de Managed Service para Apache Flink que utilice Apache Beam, consulte [Uso de Apache Beam](#).

Especificar la versión de Apache Flink de su aplicación

Al utilizar la versión 1.1.0 del tiempo de ejecución de Managed Service para Apache Flink y versiones posteriores, debe especificar la versión de Apache Flink que utilizará la aplicación al compilarla. Debe proporcionar la versión de Apache Flink con el parámetro `-Dflink.version` siguiente manera:

```
mvn package -Dflink.version=1.15.3
```

Para crear aplicaciones con versiones anteriores de Apache Flink, consulte [Versiones anteriores](#).

Creación de su aplicación de Managed Service para Apache Flink

Una vez que haya creado el código de la aplicación, haga lo siguiente para crear su aplicación de Managed Service para Apache Flink:

- **Cargue el código de la aplicación:** cargue el código de la aplicación en un bucket de Amazon S3. Al crear la aplicación, especifique el nombre del bucket de S3 y el nombre del objeto del código de la aplicación. Para ver un tutorial que muestra cómo cargar el código de la aplicación, consulte [the section called “Cargar el código de Java de streaming de Apache Flink”](#) en el tutorial [Cómo empezar \(DataStream API\)](#).
- **Cree su aplicación de Managed Service para Apache Flink:** utilice uno de los siguientes métodos para crear su aplicación de Managed Service para Apache Flink:
 - Cree su aplicación de Managed Service para Apache Flink mediante la consola de AWS: puede crear y configurar la aplicación mediante la consola de AWS.

Al crear la aplicación mediante la consola, se crean automáticamente los recursos dependientes de la aplicación (como los CloudWatch registros, las transmisiones, las funciones de IAM y las políticas de IAM).

Al crear la aplicación mediante la consola, debe especificar qué versión de Apache Flink utiliza la aplicación seleccionándola en el menú desplegable de la página Managed Service para Apache Flink: Crear aplicación.

Para ver un tutorial acerca de cómo utilizar la consola para crear una aplicación, consulte [the section called “Crear y ejecutar la aplicación \(consola\)”](#) en el tutorial [Cómo empezar \(DataStream API\)](#).

- Cree su aplicación de Managed Service para Apache Flink mediante la CLI de AWS: puede crear y configurar la aplicación mediante la CLI de AWS.

Al crear la aplicación mediante la CLI, también debe crear los recursos dependientes de la aplicación (como las transmisiones de CloudWatch registros, las funciones de IAM y las políticas de IAM) de forma manual.

Al crear la aplicación mediante la CLI, debe especificar qué versión de Apache Flink utiliza la aplicación mediante el parámetro `RuntimeEnvironment` de la acción `CreateApplication`.

Para ver un tutorial acerca de cómo utilizar la CLI para crear una aplicación, consulte [the section called “Crear y ejecutar una aplicación mediante la CLI”](#) en el tutorial [Cómo empezar \(DataStream API\)](#).

Note

No puede cambiar el `RuntimeEnvironment` de una aplicación existente. Si necesita cambiar el `RuntimeEnvironment` de una aplicación existente, debe eliminarla y volver a crearla.

Iniciar su aplicación de Managed Service para Apache Flink

Una vez que haya creado el código de la aplicación, lo haya cargado en S3 y creado la aplicación Managed Service para Apache Flink, inicie la aplicación. El inicio de una aplicación Managed Service para Apache Flink normalmente tarda varios minutos.

Utilice uno de los siguientes métodos para iniciar la aplicación:

- Inicie la aplicación Managed Service para Apache Flink mediante la consola de AWS: puede ejecutar la aplicación seleccionando Ejecutar en la página de la aplicación, en la consola de AWS.
- Inicie su aplicación Managed Service for Apache Flink mediante la AWS API: puede ejecutar su aplicación mediante la acción. [StartApplication](#)

Verificación de su aplicación Managed Service para Apache Flink

Puede comprobar que la aplicación funciona de las siguientes maneras:

- Uso de CloudWatch registros: puede usar CloudWatch Logs y CloudWatch Logs Insights para comprobar que la aplicación se ejecuta correctamente. Para obtener información sobre el uso de CloudWatch Logs con su aplicación Managed Service for Apache Flink, consulte [Registro y supervisión](#).
- Uso de CloudWatch métricas: puede utilizar CloudWatch las métricas para supervisar la actividad de la aplicación o la actividad de los recursos que la aplicación utiliza como entrada o salida (como las transmisiones de Kinesis, las transmisiones de Kinesis Data Firehose o los buckets de Amazon S3). Para obtener más información sobre CloudWatch las métricas, consulta [Cómo trabajar con métricas](#) en la Guía del CloudWatch usuario de Amazon.
- Supervisión de las ubicaciones de salida: si la aplicación escribe la salida en una ubicación (como un bucket o una base de datos de Amazon S3), puede supervisar esa ubicación para localizar los datos escritos.

Ejecución de un aplicación de Managed Service para Apache Flink

En este tema, se incluye información acerca de cómo ejecutar una aplicación de Managed Service para Apache Flink.

Al ejecutar la aplicación de Managed Service para Apache Flink, el servicio crea un trabajo de Apache Flink. Un trabajo de Apache Flink es el ciclo de vida de ejecución de la aplicación de Managed Service para Apache Flink. El administrador de trabajos administra la ejecución del trabajo y los recursos que utiliza. El administrador de trabajos divide la ejecución de la aplicación en tareas. Cada tarea es gestionada por un administrador de tareas. Al supervisar el rendimiento de la aplicación, se puede examinar el rendimiento de cada administrador de tareas o del administrador de trabajos en su conjunto.

Para obtener más información sobre trabajos en Apache Flink, consulte [Trabajos y programación](#) en la [documentación de Apache Flink](#).

Solicitud y estado del trabajo

Tanto su solicitud como el trabajo de la solicitud tienen un estado de ejecución actual:

- Estado de la solicitud: su solicitud tiene un estado actual que describe su fase de ejecución. El estado de la aplicación puede ser cualquiera de los siguientes:
 - Estado de solicitud estable: la solicitud normalmente permanece en este estado hasta que se realiza un cambio de estado:

- **LISTA**: una aplicación nueva o detenida se encuentra LISTA hasta que se la ejecuta.
- **EN EJECUCIÓN**: una aplicación que se ha iniciado correctamente se encuentra EN EJECUCIÓN.
- **Estado de aplicación transitorio**: una aplicación en este estado suele estar en proceso de transición a otro estado. Si la aplicación permanece en un estado transitorio por mucho tiempo, puede detenerla mediante la acción [StopApplication](#) con el parámetro Force establecido en true. Este estado incluye los siguientes:
 - **STARTING**: se produce después de la acción [StartApplication](#). La aplicación está pasando del estado READY al RUNNING.
 - **STOPPING**: se produce después de la acción [StopApplication](#). La aplicación está pasando del estado RUNNING al READY.
 - **DELETING**: se produce después de la acción [DeleteApplication](#). La aplicación está en proceso de ser eliminada.
 - **UPDATING**: se produce después de la acción [UpdateApplication](#). La aplicación se está actualizando y volverá al estado RUNNING o READY.
 - **AUTOSCALING**: la aplicación tiene la propiedad AutoScalingEnabled de [ParallelismConfiguration](#) establecida en true y el servicio aumenta el paralelismo de la aplicación. Cuando la aplicación se encuentra en este estado, la única acción de API válida que se puede usar es la acción [StopApplication](#) con el parámetro Force establecido en true. Para obtener información sobre el escalado automático, consulte [Escalado automático](#).
 - **FORCE_STOPPING**: se produce después de solicitar la acción [StopApplication](#) con el parámetro Force establecido en true. La aplicación está en proceso de ser detenida forzosamente. La aplicación está pasando del estado STARTING, UPDATING, STOPPING o AUTOSCALING al READY.
 - **ROLLING_BACK**: se produce después de solicitar la acción [RollbackApplication](#). La aplicación está en proceso de revertirse a una versión anterior. La aplicación está pasando del estado UPDATING o AUTOSCALING al RUNNING.
 - **ROLLED_BACK**: cuando se revierte correctamente una aplicación, este pasa a ser el estado de la versión desde su reversión. Para obtener información sobre cómo revertir una aplicación, consulte [RollbackApplication](#).
 - **MAINTENANCE**: se produce mientras Managed Service para Apache Flink aplica parches a la aplicación. Para obtener más información, consulte [Mantenimiento](#).

Puede comprobar el estado de la aplicación mediante la consola o mediante la acción

[DescribeApplication](#).

- Estado del trabajo: cuando la solicitud está en el estado RUNNING, el trabajo tiene un estado que describe su fase de ejecución actual. El trabajo comienza en el estado CREATEDy, a continuación, pasa al estado RUNNING cuando se inicia. Si se producen condiciones de error, la aplicación pasa al siguiente estado:
 - En el caso de las aplicaciones que utilizan Apache Flink 1.11 y versiones posteriores, la aplicación ingresa al estado RESTARTING.
 - En el caso de las aplicaciones que utilizan Apache Flink 1.8 y versiones anteriores, la aplicación ingresa al estado FAILING.

A continuación, la aplicación pasa al estado RESTARTING o FAILED, en función de si se puede reiniciar el trabajo.

Para comprobar el estado del trabajo, verifique el registro de CloudWatch de la aplicación para ver si hubo cambios de estado.

Cargas de trabajo en lotes

Managed Service para Apache Flink admite la ejecución de cargas de trabajo por lotes de Apache Flink. En un trabajo por lotes, cuando un trabajo de Apache Flink pasa al estado FINALIZADO, el estado de la aplicación de Managed Service para Apache Flink se establece en LISTO. Para obtener más información sobre el estado de los trabajos de Flink, consulte [Trabajos y programación](#).

Recursos de aplicaciones

En esta sección se describen los recursos del sistema que utiliza la aplicación. Comprender cómo Managed Service for Apache Flink aprovisiona y utiliza los recursos le ayudará a diseñar, crear y mantener una aplicación Managed Service para Apache Flink estable y de alto rendimiento.

Recursos de aplicaciones de Managed Service para Apache Flink

Managed Service para Apache Flink es un AWS servicio que crea un entorno para alojar su aplicación Apache Flink. El servicio Managed Service para Apache Flink proporciona recursos mediante unidades denominadas Unidades de procesamiento de Kinesis (KPU).

Una KPU representa los siguientes recursos del sistema:

- Un núcleo de CPU
- 4 GB de memoria, de los cuales un GB es memoria nativa y tres GB son memoria de pila
- 50 GB de espacio en disco

Las KPU ejecutan las aplicaciones en distintas unidades de ejecución denominadas tareas y subtareas. Puede pensar en una subtarea como el equivalente a un hilo.

El número de KPU disponibles para una aplicación es igual a la `Parallelism` configuración de la aplicación dividida por la configuración de `ParallelismPerKPU` la aplicación.

Para obtener más información acerca del paralelismo de las aplicaciones, consulte [Escalado](#).

Recursos de la aplicación de Apache Flink

El entorno Apache Flink asigna los recursos a su aplicación mediante unidades denominadas ranuras de tareas. Cuando Managed Service para Apache Flink asigna recursos a su aplicación, asigna uno o más ranuras de tareas de Apache Flink a una sola KPU. La cantidad de ranuras asignadas a una sola KPU es igual a la configuración de su aplicación `ParallelismPerKPU`. Para obtener más información acerca de las ranuras de tareas, consulte [Programación de trabajos](#) en la [documentación de Apache Flink](#).

Paralelismo del operador

Puede establecer el número máximo de subtareas que puede utilizar un operador. Este valor se denomina Paralelismo del operador. De forma predeterminada, el paralelismo de cada operador de su aplicación es igual al paralelismo de la aplicación. Esto significa que, de forma predeterminada, cada operador de su aplicación puede usar todas las subtareas disponibles en la aplicación si es necesario.

Puede establecer el paralelismo de los operadores de su aplicación mediante el método `setParallelism`. Con este método, puede controlar la cantidad de subtareas que cada operador puede usar a la vez.

Para obtener más información sobre el encadenamiento de operadores, consulte [Encadenamiento de tareas y grupos de recursos](#) en la documentación de [Apache Flink](#).

Encadenamiento de operadores

Normalmente, cada operador utiliza una subtarea independiente para ejecutar, pero si varios operadores se ejecutan siempre en secuencia, el tiempo de ejecución puede asignarlos todos a la misma tarea. Este proceso se denomina Encadenamiento de operadores.

Se pueden encadenar varios operadores secuenciales en una sola tarea si todos funcionan con los mismos datos. A continuación se muestran algunos criterios necesarios para que esto sea cierto:

- Los operadores realizan un enrutamiento sencillo de uno a uno.
- Todos los operadores tienen el mismo paralelismo de operadores.

Cuando su aplicación encadena a los operadores en una sola subtarea, conserva los recursos del sistema, ya que el servicio no necesita realizar operaciones de red ni asignar subtareas a cada operador. Para determinar si su aplicación utiliza el encadenamiento de operadores, consulte el gráfico de trabajo de la consola de Managed Service para Apache Flink. Cada vértice de la aplicación representa uno o más operadores. El gráfico muestra los operadores que se han encadenado como un solo vértice.

API de DataStream

Su aplicación Apache Flink utiliza la [API Apache Flink DataStream](#) para transformar los datos en un flujo de datos.

Esta sección contiene los siguientes temas:

- [Uso de conectores para mover datos en un servicio gestionado para Apache Flink con la API DataStream](#): estos componentes mueven los datos entre la aplicación y el origen y destino de datos externos.
- [Uso de operadores para transformar datos en Managed Service para Apache Flink con la API de DataStream](#): estos componentes transforman o agrupan los elementos de datos dentro de la aplicación.
- [Seguimiento de eventos en Managed Service para Apache Flink mediante la API de DataStream](#): en este tema se describe cómo Managed Service para Apache Flink realiza un seguimiento de los eventos cuando se usa la API de DataStream.

Uso de conectores para mover datos en un servicio gestionado para Apache Flink con la API DataStream

En la DataStream API de Amazon Managed Service for Apache Flink, los conectores son componentes de software que mueven datos hacia y desde una aplicación de Managed Service for Apache Flink. Los conectores son integraciones flexibles que permiten leer archivos y directorios. Los conectores constan de módulos completos para interactuar con los servicios de Amazon y los sistemas de terceros.

Entre los tipos de conectores, se incluyen:

- [Orígenes](#): proporcione datos a su aplicación desde un flujo de datos de Kinesis, un archivo u otro origen de datos.
- [Receptores](#): envíe datos desde su aplicación a un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose u otro destino de datos.
- [E/S asíncrona](#): Proporciona acceso asíncrono a un origen de datos (como una base de datos) para enriquecer los eventos de flujos.

Conectores disponibles

El marco de Apache Flink contiene conectores para acceder a los datos desde una variedad de fuentes. Para obtener información sobre los conectores disponibles en el marco Apache Flink, consulte la sección [Conectores](#) de la [documentación de Apache Flink](#).

Warning

Si tiene aplicaciones que se ejecutan en Flink 1.6, 1.8, 1.11 o 1.13 y desea ejecutarlas en las regiones de Medio Oriente (EAU), Asia Pacífico (Hyderabad), Israel (Tel Aviv), Europa (Zúrich), Medio Oriente (EAU), Asia-Pacífico (Melbourne) o Asia Pacífico (Yakarta), puede que necesite volver a compilar el archivo de su aplicación con un conector actualizado o actualizar a Flink 1.15. Las siguientes son las pautas recomendadas:

Actualizaciones de conectores

Versión de Flink	Conector utilizado	Resolución
1. - 1.	Firehose	Su aplicación depende de una versión anticuada del conector Firehose que no está al tanto de las regiones más recientes de AWS. Vuelva a compilar el archivo de su

Versión de Flink	Conector utilizado	Resolución
		aplicación con la versión 2.1.0 del conector Firehose. v2.1.0

Versión de Flink	Conector utilizado	Resolución
1.	Kinesis	Su aplicación depende de una versión anticuada del conector Kinesis de Flink que no está al tanto de las regiones más recientes de AWS. Vuelva a compilar el archivo de su

Versión de Flink	Conector utilizado	Resolución
		aplicación con la versión 1.6.1 del conector Kinesis de Flink. https://github.com/aws-labs/flink-kinesis-connector

Versión de Flink	Conector utilizado	Resolución
1.	Kinesis	Su aplicación depende de una versión anticuada del conector Kinesis de Flink que no está al tanto de las regiones más recientes de AWS. Vuelva a compilar el archivo de su

Versión de Flink	Conector utilizado	Resolución
		aplicación con la versión 2.4.1 del conector Kinesis de Flink. https://github.com/aws-labs/amazon-kinesis-connector-flink/tree/2.4.1

Versión de Flink	Conector utilizado	Resolución
1. y 1.	Kinesis	Su aplicación depende de una versión anticuada del conector Kinesis de Flink que no está al tanto de las regiones más recientes de AWS. Lamentablemente, Flink ya no publica parches

Versión de Flink	Conector utilizado	Resolución
		ninguna corrección de errores para los conectores 1.6/1.13. Le sugerimos que actualice a Flink 1.15 volviendo a compilar el archivo de su aplicación con Flink 1.15.

Agregar orígenes de datos de streaming a Managed Service para Apache Flink

Apache Flink proporciona conectores para leer archivos, sockets, colecciones y fuentes personalizadas. En el código de su aplicación, debe utilizar una [fuente de Apache Flink](#) para recibir datos de un flujo. En esta sección se describen las fuentes disponibles para los servicios de Amazon.

Kinesis Data Streams

La fuente `FlinkKinesisConsumer` proporciona datos de streaming a su aplicación desde un flujo de datos de Amazon Kinesis.

Creación de un `FlinkKinesisConsumer`

En el siguiente código de ejemplo se muestra la creación de un `FlinkKinesisConsumer`:

```
Properties inputProperties = new Properties();
inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

DataStream<string> input = env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
    new SimpleStringSchema(), inputProperties));
```

Para obtener más información sobre cómo usar un `FlinkKinesisConsumer`, consulte [Descargar y examinar el código de Java de streaming de Apache Flink](#).

Crear un `FlinkKinesisConsumer` que utilice un consumidor de EFO

`FlinkKinesisConsumer` Ahora es compatible con [Enhanced Fan-Out](#) (EFO).

Si un consumidor de Kinesis usa EFO, el servicio Kinesis Data Streams le proporciona su propio ancho de banda dedicado, en lugar de hacer que el consumidor comparta el ancho de banda fijo del flujo con los demás consumidores que leen el flujo.

Para obtener más información sobre el uso de EFO con el consumidor Kinesis, consulte [FLIP-128: distribución mejorada para consumidores de AWS Kinesis](#).

Para activar el consumidor EFO, configure los siguientes parámetros en el consumidor de Kinesis:

- `RECORD_PUBLISHER_TYPE`: defina este parámetro en EFO para que su aplicación utilice un consumidor de EFO para acceder a los datos del flujo de datos de Kinesis.

- `EFO_CONSUMER_NAME`: defina este parámetro en un valor de cadena que sea único entre los consumidores de este flujo. La reutilización de un nombre de consumidor en el mismo flujo de datos de Kinesis provocará la cancelación del consumidor anterior que utilizó ese nombre.

A fin de configurar un `FlinkKinesisConsumer` para que use EFO, añada los siguientes parámetros al consumidor:

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

Para ver un ejemplo de una aplicación de Managed Service para Apache Flink que utiliza un consumidor de EFO, consulte [Consumidor de EFO](#).

Amazon MSK

La fuente `KafkaSource` proporciona datos de streaming a su aplicación desde un tema de Amazon MSK.

Creación de un `KafkaSource`

En el siguiente código de ejemplo se muestra la creación de un `KafkaSource`:

```
KafkaSource<String> source = KafkaSource.<String>builder()
    .setBootstrapServers(brokers)
    .setTopics("input-topic")
    .setGroupId("my-group")
    .setStartingOffsets(OffsetsInitializer.earliest())
    .setValueOnlyDeserializer(new SimpleStringSchema())
    .build();

env.fromSource(source, WatermarkStrategy.noWatermarks(), "Kafka Source");
```

Para obtener más información sobre cómo usar un `KafkaSource`, consulte [Replicación MSK](#).

Escritura de datos mediante receptores en Managed Service para Apache Flink

En el código de la aplicación, utilice un [receptor de Apache Flink](#) para escribir datos de un flujo de Apache Flink en un servicio de AWS, como Kinesis Data Streams.

Apache Flink proporciona receptores para archivos, sockets y receptores personalizados. Los siguientes receptores están disponibles para AWS:

Kinesis Data Streams

Apache Flink proporciona información sobre el [conector de Kinesis Data Streams](#) en la documentación de Apache Flink.

Para ver un ejemplo de una aplicación que utiliza un flujo de datos de Kinesis como entrada y salida, consulte [Cómo empezar \(DataStream API\)](#).

Amazon S3

Puede utilizar el `StreamingFileSink` de Apache Flink para escribir objetos en un bucket de Amazon S3.

Para ver un ejemplo sobre cómo escribir objetos en S3, consulte [the section called “Receptor S3”](#).

Kinesis Data Firehose

El `FlinkKinesisFirehoseProducer` es un receptor de Apache Flink fiable y escalable para almacenar los resultados de las aplicaciones mediante el servicio [Kinesis Data Firehose](#). En esta sección se describe cómo configurar un proyecto de Maven para crear y utilizar un `FlinkKinesisFirehoseProducer`.

Temas

- [Creación de un FlinkKinesisFirehoseProducer](#)
- [Ejemplo de código de FlinkKinesisFirehoseProducer](#)

Creación de un **FlinkKinesisFirehoseProducer**

En el siguiente código de ejemplo se muestra la creación de un `FlinkKinesisFirehoseProducer`:

```
Properties outputProperties = new Properties();
outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

FlinkKinesisFirehoseProducer<String> sink = new
    FlinkKinesisFirehoseProducer<>(outputStreamName, new SimpleStringSchema(),
        outputProperties);
```

Ejemplo de código de **FlinkKinesisFirehoseProducer**

El siguiente ejemplo de código muestra cómo crear y configurar un `FlinkKinesisFirehoseProducer` y enviar datos desde un flujo de datos de Apache Flink al servicio Kinesis Data Firehose.

```
package com.amazonaws.services.kinesisanalytics;

import
    com.amazonaws.services.kinesisanalytics.flink.connectors.config.ProducerConfigConstants;
import
    com.amazonaws.services.kinesisanalytics.flink.connectors.producer.FlinkKinesisFirehoseProducer;
import com.amazonaws.services.kinesisanalytics.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;

import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

public class StreamingJob {

    private static final String region = "us-east-1";
    private static final String inputStreamName = "ExampleInputStream";
    private static final String outputStreamName = "ExampleOutputStream";

    private static DataStream<String>
    createSourceFromStaticConfig(StreamExecutionEnvironment env) {
        Properties inputProperties = new Properties();
        inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
        inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
            "LATEST");

        return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
            SimpleStringSchema(), inputProperties));
    }
}
```

```
private static DataStream<String>
createSourceFromApplicationProperties(StreamExecutionEnvironment env)
    throws IOException {
    Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
    return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
SimpleStringSchema(),
    applicationProperties.get("ConsumerConfigProperties")));
}

private static FlinkKinesisFirehoseProducer<String>
createFirehoseSinkFromStaticConfig() {
    /*
    * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
    * ProducerConfigConstants
    * lists of all of the properties that firehose sink can be configured with.
    */

    Properties outputProperties = new Properties();
    outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

    FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName,
    new SimpleStringSchema(), outputProperties);
    ProducerConfigConstants config = new ProducerConfigConstants();
    return sink;
}

private static FlinkKinesisFirehoseProducer<String>
createFirehoseSinkFromApplicationProperties() throws IOException {
    /*
    * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
    * ProducerConfigConstants
    * lists of all of the properties that firehose sink can be configured with.
    */

    Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
    FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName,
    new SimpleStringSchema(),
    applicationProperties.get("ProducerConfigProperties"));
    return sink;
}
```

```
public static void main(String[] args) throws Exception {
    // set up the streaming execution environment
    final StreamExecutionEnvironment env =
    StreamExecutionEnvironment.getExecutionEnvironment();

    /*
     * if you would like to use runtime configuration properties, uncomment the
     * lines below
     * DataStream<String> input = createSourceFromApplicationProperties(env);
     */

    DataStream<String> input = createSourceFromStaticConfig(env);

    // Kinesis Firehose sink
    input.addSink(createFirehoseSinkFromStaticConfig());

    // If you would like to use runtime configuration properties, uncomment the
    // lines below
    // input.addSink(createFirehoseSinkFromApplicationProperties());

    env.execute("Flink Streaming Java API Skeleton");
}
}
```

Para ver un tutorial completo sobre cómo utilizar el receptor de Kinesis Data Firehose, consulte [the section called “Receptor de Kinesis Data Firehose”](#).

Uso de E/S asíncrona en Managed Service para Apache Flink

Un operador de E/S asíncrona enriquece los datos del flujo mediante un origen de datos externo, como una base de datos. Managed Service para Apache Flink enriquece los eventos del flujo de forma asíncrona para que las solicitudes se puedan agrupar en lotes y aumentar la eficiencia.

Para obtener más información, consulte [E/S asíncrona](#) en la [documentación de Apache Flink](#).

Uso de operadores para transformar datos en Managed Service para Apache Flink con la API de DataStream

Para transformar los datos entrantes en un Managed Service para Apache Flink, utiliza un operador de Apache Flink. Un operador de Apache Flink transforma uno o más flujos de datos en un nuevo

flujo de datos. El nuevo flujo de datos contiene datos modificados del flujo de datos original. Apache Flink proporciona más de 25 operadores de procesamiento de flujos prediseñados. Para obtener más información, consulte [Operadores](#) en la [documentación de Apache Flink](#).

Este tema contiene las siguientes secciones:

- [Transformación de operadores](#)
- [Operadores de agregación](#)

Transformación de operadores

El siguiente es un ejemplo de una transformación de texto simple en uno de los campos de un flujo de datos JSON.

Este código crea un flujo de datos transformado. El nuevo flujo de datos tiene los mismos datos que el flujo original, con la cadena “ Company” anexada al contenido del campo TICKER.

```
DataStream<ObjectNode> output = input.map(
    new MapFunction<ObjectNode, ObjectNode>() {
        @Override
        public ObjectNode map(ObjectNode value) throws Exception {
            return value.put("TICKER", value.get("TICKER").asText() + " Company");
        }
    }
);
```

Operadores de agregación

A continuación se muestra un ejemplo de operador de agregación. El código crea un flujo de datos agregado. El operador crea una ventana de caída de 5 segundos y devuelve la suma de los valores PRICE de los registros de la ventana con el mismo valor TICKER.

```
DataStream<ObjectNode> output = input.keyBy(node -> node.get("TICKER").asText())
    .window(TumblingProcessingTimeWindows.of(Time.seconds(5)))
    .reduce((node1, node2) -> {
        double priceTotal = node1.get("PRICE").asDouble() +
node2.get("PRICE").asDouble();
        node1.replace("PRICE", JsonNodeFactory.instance.numberNode(priceTotal));
        return node1;
    });
```


Para ver un ejemplo de código completo que utiliza operadores, consulte [Cómo empezar \(DataStream API\)](#). El código fuente de la aplicación Getting Started está disponible en [Introducción](#), en el repositorio GitHub de [Ejemplos de Java de Managed Service para Apache Flink](#).

Seguimiento de eventos en Managed Service para Apache Flink mediante la API de DataStream

Managed Service para Apache Flink realiza un seguimiento de los eventos mediante las siguientes marcas de tiempo:

- Hora de procesamiento: se refiere a la hora del sistema de la máquina que está ejecutando la operación correspondiente.
- Hora del evento: se refiere a la hora en que ocurrió cada evento individual en el dispositivo que lo produjo.
- Hora de adquisición de datos: se refiere al momento en que los eventos ingresan al servicio Managed Service para Apache Flink.

Usted establece la hora utilizada por el entorno de streaming mediante [setStreamTimeCharacteristic](#):

```
env.setStreamTimeCharacteristic(TimeCharacteristic.ProcessingTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.IngestionTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);
```

Para obtener más información sobre marcas de tiempo, consulte la [Hora del evento](#) en la [Documentación de Apache Flink](#).

API de tabla

Su aplicación Apache Flink usa la [API de tabla de Apache Flink](#) para interactuar con los datos de un flujo mediante un modelo relacional. La API de tabla se usa para acceder a los datos mediante fuentes de tablas y, luego, las funciones de tabla se usan para transformar y filtrar los datos de las tablas. Los datos tabulares se pueden transformar y filtrar mediante funciones de la API, o bien, comandos SQL.

Esta sección contiene los siguientes temas:

- [Conectores de la API de tabla](#): estos componentes mueven los datos entre la aplicación y el origen y destino de datos externos.
- [Atributos de tiempo de la API de tabla](#): este tema describe cómo Managed Service para Apache Flink realiza un seguimiento de los eventos cuando se usa la API de tabla.

Conectores de la API de tabla

En el modelo de programación de Apache Flink, los conectores son componentes que la aplicación utiliza para leer o escribir datos de fuentes externas, como otros servicios de AWS.

Con la API de tabla de Apache Flink, se pueden usar los siguientes tipos de conectores:

- [Fuentes de la API de tabla](#): se utilizan los conectores fuente de la API de tabla para crear tablas dentro de su `TableEnvironment` mediante solicitudes a la API o, bien, consultas SQL.
- [Receptores de la API de tabla](#): se utilizan comandos SQL para escribir datos de tablas en fuentes externas, como un tema de Amazon MSK o un bucket de Amazon S3.

Fuentes de la API de tabla

Se crea una fuente de tabla a partir de un flujo de datos. El siguiente código crea una tabla a partir de un tema de Amazon MSK:

```
//create the table
    final FlinkKafkaConsumer<StockRecord> consumer = new
FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
kafkaProperties);
    consumer.setStartFromEarliest();
    //Obtain stream
    DataStream<StockRecord> events = env.addSource(consumer);

    Table table = streamTableEnvironment.fromDataStream(events);
```

Para obtener más información sobre fuentes de tablas, consulte [Tabla y conectores](#) en la [documentación de Apache Flink](#).

Receptores de la API de tabla

Para escribir datos de tabla en un receptor, se crea el receptor en SQL y, a continuación, se ejecuta el receptor basado en SQL en el objeto `StreamTableEnvironment`.

En el siguiente código de ejemplo, se muestra cómo escribir datos de tablas a un receptor de Amazon S3:

```
final String s3Sink = "CREATE TABLE sink_table (" +
    "event_time TIMESTAMP," +
    "ticker STRING," +
    "price DOUBLE," +
    "dt STRING," +
    "hr STRING" +
    ")" +
    " PARTITIONED BY (ticker,dt,hr)" +
    " WITH" +
    "(" +
    " 'connector' = 'filesystem'," +
    " 'path' = '" + s3Path + "'," +
    " 'format' = 'json'" +
    ") ";

//send to s3
streamTableEnvironment.executeSql(s3Sink);
filteredTable.executeInsert("sink_table");
```

Puede usar el parámetro `format` para controlar el formato que Managed Service para Apache Flink utiliza para escribir el resultado en el receptor. Para obtener información sobre formatos, consulte [Formatos](#) en la [documentación de Apache Flink](#).

Para obtener más información sobre receptores de tablas, consulte [Tabla y conectores](#) en la [documentación de Apache Flink](#).

Orígenes y receptores definidos por el usuario

Puede usar los conectores Apache Kafka existentes para enviar datos a otros servicios de AWS desde estos, como Amazon MSK y Amazon S3. Para interactuar con otros orígenes y destinos de datos, puede definir sus propios orígenes y receptores. Para obtener más información, consulte [Orígenes y receptores definidos por el usuario](#) en la [documentación de Apache Flink](#).

Atributos de tiempo de la API de tabla

Cada registro de un flujo de datos tiene varias marcas de tiempo que definen cuándo ocurrieron los eventos relacionados con el registro:

- Hora del evento: una marca de tiempo definida por el usuario que define cuándo ocurrió el evento que creó el registro.
- Hora de adquisición de datos: la hora en que la aplicación obtuvo el registro del flujo de datos.
- Hora de procesamiento: la hora en que su solicitud procesó el registro.

Cuando la API de tabla de Apache Flink crea ventanas en función de los tiempos de registro, usted define cuáles de estas marcas de tiempo utiliza con el método [setStreamTimeCharacteristic](#).

Para obtener más información sobre el uso de marcas de tiempo con la API de tabla, consulte los [Atributos de tiempo](#) en la [documentación de Apache Flink](#).

Uso de Python con Managed Service para Apache Flink

Note

Si está desarrollando la aplicación Python Flink en una Mac nueva con el chip Apple Silicon, es posible que experimente ciertos [problemas conocidos](#) con las dependencias de Python de PyFlink 1.15. En este caso, recomendamos ejecutar el intérprete de Python en Docker. Para obtener instrucciones paso a paso, consulte [Desarrollo de PyFlink 1.15 en la Mac con Apple Silicon](#).

La versión 1.15.2 de Apache Flink incluye soporte para crear aplicaciones con Python versión 3.8, utilizando la biblioteca [PyFlink](#). Para crear una aplicación de Managed Service para Apache Flink mediante Python, haga lo siguiente:

- Cree el código de su aplicación de Python como un archivo de texto con un método `main`.
- Agrupe el archivo de código de la aplicación y cualquier dependencia de Python o Java en un archivo zip y cárguelo en un bucket de Amazon S3.
- Cree su aplicación Managed Service para Apache Flink especificando la ubicación del código de Amazon S3, las propiedades y la configuración de la aplicación.

En un nivel alto, la API Python Table es un envoltorio alrededor de la API Java Table. Para obtener información sobre la API de tabla de Python, consulte [Introducción a la API de tabla de Python](#) en la documentación de [Apache Flink](#).

Programar una aplicación de Managed Service para Apache Flink para Python

Usted codifica su aplicación Managed Service para Apache Flink para Python mediante la API Apache Flink Python Table. El motor Apache Flink traduce las declaraciones de la API de tablas de Python (que se ejecutan en la máquina virtual de Python) en declaraciones de la API de tabla de Java (que se ejecutan en la máquina virtual de Java).

Para utilizar la Python Table API, siga estos pasos:

- Cree una referencia a `StreamTableEnvironment`.
- Cree `table` objetos a partir de sus datos de transmisión de origen ejecutando consultas en la referencia `StreamTableEnvironment`.
- Ejecute consultas en sus objetos `table` para crear tablas de salida.
- Escriba sus tablas de salida en sus destinos utilizando un `StatementSet`.

Para empezar a utilizar la API de tabla de Python en Managed Service para Apache Flink, consulte [Introducción a Amazon Managed Service para Apache Flink para Python](#).

Lectura y escritura de datos de streaming

Para leer y escribir datos de streaming, ejecute consultas SQL en el entorno de tablas.

Creación de una tabla

El siguiente ejemplo de código muestra una función definida por el usuario que crea una consulta SQL. La consulta SQL crea una tabla que interactúa con un flujo de Kinesis:

```
def create_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
        `record_id` VARCHAR(64) NOT NULL,
        `event_time` BIGINT NOT NULL,
        `record_number` BIGINT NOT NULL,
        `num_retries` BIGINT NOT NULL,
        `verified` BOOLEAN NOT NULL
    )
    PARTITIONED BY (record_id)
    WITH (
```

```
'connector' = 'kinesis',
'stream' = '{1}',
'aws.region' = '{2}',
'scan.stream.initpos' = '{3}',
'sink.partitioner-field-delimiter' = ';',
'sink.producer.collection-max-count' = '100',
'format' = 'json',
'json.timestamp-format.standard' = 'ISO-8601'
) """.format(table_name, stream_name, region, stream_initpos)
```

Lectura de datos de streaming

El siguiente ejemplo de código muestra cómo utilizar la consulta SQL CreateTable anterior en una referencia de entorno de tabla para leer datos:

```
table_env.execute_sql(create_table(input_table, input_stream, input_region,
stream_initpos))
```

Escribir datos de streaming

El siguiente ejemplo de código muestra cómo utilizar la consulta SQL del ejemplo CreateTable para crear una referencia a la tabla de salida y cómo utilizar un StatementSet para interactuar con las tablas y escribir datos en un flujo de Kinesis de destino:

```
table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
    .format(output_table_name, input_table_name))
```

Leer propiedades de tiempo de ejecución

Puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que cambiar el código de la aplicación.

Las propiedades de la aplicación se especifican de la misma manera que en el caso de una aplicación de Managed Service para Apache Flink para Java. Puede especificar propiedades del tiempo de ejecución de las siguientes maneras:

- Usando la acción [CreateApplication](#).
- Usando la acción [UpdateApplication](#).
- Configurar su aplicación usando la consola.

Puede recuperar las propiedades de la aplicación en el código leyendo un archivo json llamado `application_properties.json`, creado por el motor de ejecución Managed Service para Apache Flink.

El siguiente ejemplo de código demuestra las propiedades de aplicación de lectura desde el archivo `application_properties.json`:

```
file_path = '/etc/flink/application_properties.json'
if os.path.isfile(file_path):
    with open(file_path, 'r') as file:
        contents = file.read()
        properties = json.loads(contents)
```

El siguiente ejemplo de código de función definido por el usuario muestra la lectura de un grupo de propiedades del objeto de propiedades de la aplicación: recupera:

```
def property_map(properties, property_group_id):
    for prop in props:
        if prop["PropertyGroupId"] == property_group_id:
            return prop["PropertyMap"]
```

En el siguiente ejemplo de código, se muestra la lectura de una propiedad denominada `INPUT_STREAM_KEY` de un grupo de propiedades que se devuelve en el ejemplo anterior:

```
input_stream = input_property_map[INPUT_STREAM_KEY]
```

Crear el paquete de códigos de la aplicación

Una vez que haya creado su aplicación Python, agrupe el archivo de código y las dependencias en un archivo zip.

El archivo zip debe contener una secuencia de comandos de Python con un método `main` y, de forma opcional, puede contener lo siguiente:

- Archivos de código Python adicionales
- Código Java definido por el usuario en archivos JAR
- Bibliotecas Java en archivos JAR

Note

El archivo zip de la aplicación debe contener todas las dependencias de la aplicación. No puede hacer referencia a bibliotecas de otros orígenes para su aplicación.

Creación de su aplicación de Managed Service para Apache Flink Python

Especificar sus archivos de código

Una vez que se crea el paquete de código de la aplicación, se carga a un bucket de Amazon S3. A continuación, se crea la aplicación mediante la consola o la acción [CreateApplication](#).

Al crear la aplicación mediante la acción [CreateApplication](#), se especifican los archivos de código y los archivos zip mediante un grupo de propiedades de aplicación especial denominado `kinesis.analytics.flink.run.options`. Puede definir los siguientes tipos de archivos:

- `python`: archivo de texto que contiene un método principal de Python.
- `jarfile`: archivo JAR de Java que contiene funciones de Java definidas por el usuario.
- `PyFiles`: un archivo de recursos de Python que contiene los recursos que utilizará la aplicación.
- `PyArchives`: un archivo zip que contiene archivos de recursos para la aplicación.

Para obtener más información sobre los tipos de archivos de código Python de Apache Flink, consulte [Uso de la línea de comandos](#) en la documentación de [Apache Flink](#).

Note

Managed Service para Apache Flink no admite los tipos de archivo `pyModule`, `pyExecutable`, o `pyRequirements`. Todo el código, los requisitos y las dependencias deben estar en el archivo zip. No puede especificar las dependencias que se instalarán mediante `pip`.

El siguiente fragmento json de ejemplo muestra cómo especificar las ubicaciones de los archivos dentro del archivo zip de la aplicación:

```
"ApplicationConfiguration": {
  "EnvironmentProperties": {
```



```

"PropertyGroups": [
  {
    "PropertyGroupId": "kinesis.analytics.flink.run.options",
    "PropertyMap": {
      "python": "MyApplication/main.py",
      "jarfile": "MyApplication/lib/myJarFile.jar",
      "pyFiles": "MyApplication/lib/myDependentFile.py",
      "pyArchives": "MyApplication/lib/myArchive.zip"
    }
  },

```

Supervisión de su aplicación de Python de Managed Service para Apache Flink

Puede utilizar el registro de CloudWatch de su aplicación para supervisar su aplicación Python de Managed Service para Apache Flink.

Managed Service para Apache Flink registra los siguientes mensajes para las aplicaciones Python:

- Mensajes escritos en la consola utilizando `print()` en el método `main` de la aplicación.
- Mensajes enviados en funciones definidas por el usuario mediante el paquete `logging`. El siguiente ejemplo de código muestra cómo escribir en el registro de la aplicación desde una función definida por el usuario:

```

import logging

@udf(input_types=[DataTypes.BIGINT()], result_type=DataTypes.BIGINT())
def doNothingUdf(i):
    logging.info("Got {} in the doNothingUdf".format(str(i)))
    return i

```

- Mensajes de error emitidos por la aplicación.

Si la aplicación lanza una excepción en la función `main`, aparecerá en los registros de la aplicación.

El siguiente ejemplo muestra una entrada de registro para una excepción lanzada desde el código Python:

```

2021-03-15 16:21:20.000 ----- Python Process Started
-----

```

```
2021-03-15 16:21:21.000 Traceback (most recent call last):
2021-03-15 16:21:21.000   " File ""/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py"", line 101, in
  <module>"
2021-03-15 16:21:21.000         main()
2021-03-15 16:21:21.000   " File ""/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py"", line 54, in main"
2021-03-15 16:21:21.000   "     table_env.register_function("doNothingUdf",
doNothingUdf)"
2021-03-15 16:21:21.000 NameError: name 'doNothingUdf' is not defined
2021-03-15 16:21:21.000 ----- Python Process Exited
-----
2021-03-15 16:21:21.000 Run python process failed
2021-03-15 16:21:21.000 Error occurred when trying to start the job
```

Note

Debido a problemas de rendimiento, le recomendamos que solo utilice mensajes de registro personalizados durante el desarrollo de la aplicación.

Consulta de registros con CloudWatch Insights

La siguiente consulta de CloudWatch Insights busca los registros creados por el punto de entrada de Python mientras se ejecuta la función principal de la aplicación:

```
fields @timestamp, message
| sort @timestamp asc
| filter logger like /PythonDriver/
| limit 1000
```

Propiedades de tiempo de ejecución en Service para Apache Flink

Puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.

Este tema contiene las siguientes secciones:

- [Trabajar con las propiedades de tiempo de ejecución en la CLI](#)
- [Trabajar con las propiedades de tiempo de ejecución en la consola](#)
- [Acceder a propiedades de tiempo de ejecución en un servicio gestionado para la aplicación Apache Flink](#)

Trabajar con las propiedades de tiempo de ejecución en la CLI

Puede añadir, actualizar o eliminar propiedades de tiempo de ejecución de su aplicación Managed Service for Apache Flink mediante la consola.

Note

No puede añadir propiedades de tiempo de ejecución al crear una aplicación en la consola de Managed Service para Apache Flink.

Actualización de propiedades de tiempo de ejecución para un servicio gestionado para la aplicación Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>.
2. Elija su Managed Service para la aplicación Apache Flink. Elija Detalles de la aplicación.
3. En la página de su aplicación, elija Configurar.
4. Amplíe la sección Propiedades.
5. Utilice los controles de la sección Propiedades para definir un grupo de propiedades con pares de valor clave. Utilice estos controles para añadir, actualizar o eliminar grupos de propiedades y propiedades de tiempo de ejecución.
6. Elija Actualizar.

Trabajar con las propiedades de tiempo de ejecución en la consola

Puede agregar, actualizar o eliminar propiedades en tiempo de ejecución mediante [AWS CLI](#).

En esta sección se incluyen ejemplos de solicitudes de acciones de API para configurar las propiedades de tiempo de ejecución de una aplicación. Para obtener información sobre cómo utilizar

un archivo JSON como entrada para una acción de API, consulte [Ejemplo de código de API de Managed Service para Apache Flink](#).

Note

Sustituya el ID de cuenta de muestra (*012345678901*) en los siguientes ejemplos por su ID de cuenta.

Añadir propiedades de tiempo de ejecución al crear una aplicación

El siguiente ejemplo de solicitud para la [CreateApplication](#) acción agrega dos grupos de propiedades de tiempo de ejecución (ProducerConfigProperties y ConsumerConfigProperties) al crear una aplicación:

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "java-getting-started-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "flink.stream.initpos": "LATEST",
            "aws.region": "us-west-2",
            "AggregationEnabled": "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
```

```
        "PropertyMap" : {
            "aws.region" : "us-west-2"
        }
    ]
}
}
```

Añadir y actualizar propiedades de tiempo de ejecución en una aplicación existente

El siguiente ejemplo de solicitud de la acción [UpdateApplication](#) agrega o actualiza las propiedades de tiempo de ejecución de una aplicación existente:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```

Note

Si usa una clave que no tiene una propiedad de tiempo de ejecución correspondiente en un grupo de propiedades, Managed Service para Apache Flink agrega el par de valor clave como una nueva propiedad. Si usa una clave para una propiedad de tiempo de ejecución existente en un grupo de propiedades, Managed Service para Apache Flink actualiza el valor de la propiedad.

Eliminar propiedades de tiempo de ejecución

El siguiente ejemplo de solicitud de la acción [UpdateApplication](#) elimina todas las propiedades de tiempo de ejecución y grupos de propiedades de una aplicación existente:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 3,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": []
    }
  }
}
```

Important

Si omite un grupo de propiedades existente o una clave de propiedad existente en un grupo de propiedades, ese grupo de propiedades o propiedad se elimina.

Acceder a propiedades de tiempo de ejecución en un servicio gestionado para la aplicación Apache Flink

Las propiedades de tiempo de ejecución se recuperan en el código de la aplicación Java mediante el método estático `KinesisAnalyticsRuntime.getApplicationProperties()`, que devuelve un objeto `Map<String, Properties>`.

En el siguiente ejemplo de código Java, se recuperan las propiedades de tiempo de ejecución de su aplicación:

```
Map<String, Properties> applicationProperties =  
KinesisAnalyticsRuntime.getApplicationProperties();
```

Puede recuperar un grupo de propiedades (como un objeto `Java.Util.Properties`) de la siguiente manera:

```
Properties consumerProperties = applicationProperties.get("ConsumerConfigProperties");
```

Por lo general, se configura una fuente o un receptor de Apache Flink pasando el objeto `Properties` sin necesidad de recuperar las propiedades individuales. El siguiente ejemplo de código muestra cómo crear una fuente de Flink pasando un objeto `Properties` recuperado de las propiedades de tiempo de ejecución:

```
private static FlinkKinesisProducer<String> createSinkFromApplicationProperties()  
throws IOException {  
    Map<String, Properties> applicationProperties =  
KinesisAnalyticsRuntime.getApplicationProperties();  
    FlinkKinesisProducer<String> sink = new FlinkKinesisProducer<String>(new  
SimpleStringSchema(),  
        applicationProperties.get("ProducerConfigProperties"));  
  
    sink.setDefaultStream(outputStreamName);  
    sink.setDefaultPartition("0");  
    return sink;  
}
```

Para ver un ejemplo de código completo que utiliza propiedades de tiempo de ejecución, consulte [Cómo empezar \(DataStream API\)](#). El código fuente de la aplicación Getting Started está disponible en [Introducción](#), en el repositorio GitHub de [Ejemplos de Java de Managed Service para Apache Flink](#).

Implementación de la tolerancia a errores en Managed Service para Apache Flink

Los puntos de control son el método que se utiliza para implementar la tolerancia a errores en Amazon Managed Service para Apache Flink. Un punto de control es una copia de seguridad actualizada de una aplicación en ejecución que se utiliza para recuperarse inmediatamente de una interrupción inesperada de la aplicación o de una conmutación por error.

Para obtener más información sobre los puntos de control en las aplicaciones de Apache Flink, consulte [Puntos de control](#) en la [documentación de Apache Flink](#).

Una instantánea es una copia de seguridad del estado de la aplicación creada y gestionada manualmente. Las instantáneas permiten restaurar la aplicación a un estado anterior mediante una llamada [UpdateApplication](#). Para obtener más información, consulte [Administración de copias de seguridad de aplicaciones mediante instantáneas](#).

Si la aplicación tiene habilitados los puntos de control, el servicio ofrece tolerancia a errores al crear y cargar copias de seguridad de los datos de la aplicación en caso de que la aplicación se reinicie inesperadamente. Estos reinicios inesperados de las aplicaciones pueden deberse a reinicios inesperados de trabajos, errores en las instancias, etc. Esto proporciona a la aplicación la misma semántica que cuando se ejecuta sin errores durante estos reinicios.

Si las instantáneas están habilitadas para la aplicación y se configuran mediante la configuración [ApplicationRestoreConfiguration](#) de la aplicación, el servicio proporciona una semántica de procesamiento de una sola vez durante las actualizaciones de la aplicación o durante el escalado o el mantenimiento relacionados con el servicio.

Configuración de los puntos de control en Managed Service para Apache Flink

Puede configurar el comportamiento de los puntos de control de la aplicación. Puede definir si mantiene el estado de los puntos de control, con qué frecuencia guarda su estado en los puntos de control y el intervalo mínimo entre el final de una operación de punto de control y el comienzo de otra.

Los siguientes ajustes se configuran mediante las operaciones de la API [CreateApplication](#) o [UpdateApplication](#):

- `CheckpointingEnabled` — Indica si los puntos de control están habilitados en la aplicación.
- `CheckpointInterval` — Contiene el tiempo en milisegundos entre las operaciones de los puntos de control (persistencia).
- `ConfigurationType` — Establezca este valor en `DEFAULT` para utilizar el comportamiento predeterminado de los puntos de control. Establezca este valor en `CUSTOM` para configurar otros valores.

Note

El comportamiento predeterminado de los puntos de control es el siguiente:

- CheckpointingEnabled: true
- CheckpointInterval: 60000
- MinPauseBetweenCheckpoints: 5000

Si ConfigurationType se establece en DEFAULT, se utilizarán los valores anteriores, aunque se establezcan en otros valores utilizando la AWS Command Line Interface o estableciendo los valores en el código de la aplicación.

Note

A partir de la versión 1.15 de Flink, Managed Service para Apache Flink utilizará `stop-with-savepoint` durante la creación automática de instantáneas, es decir, durante la actualización, el escalado o la detención de la aplicación.

- `MinPauseBetweenCheckpoints` — El tiempo mínimo en milisegundos entre el final de una operación de punto de control y el inicio de otra. Establecer este valor impide que la aplicación realice un punto de control de forma continua cuando una operación de punto de control tarda más de `CheckpointInterval`.

Ejemplos de la API de puntos de control

En esta sección se incluyen ejemplos de solicitudes de acciones de API para configurar los puntos de control de una aplicación. Para obtener información sobre cómo utilizar un archivo JSON como entrada para una acción de API, consulte [Ejemplo de código de API de Managed Service para Apache Flink](#).

Configurar puntos de control para una nueva aplicación

La siguiente solicitud de ejemplo de la acción [CreateApplication](#) configura los puntos de control al crear una aplicación:

```
{  
  "ApplicationName": "MyApplication",
```

```

"RuntimeEnvironment": "FLINK-1_15",
"ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
"ApplicationConfiguration": {
  "ApplicationCodeConfiguration": {
    "CodeContent": {
      "S3ContentLocation": {
        "BucketARN": "arn:aws:s3:::mybucket",
        "FileKey": "myflink.jar",
        "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    },
    "FlinkApplicationConfiguration": {
      "CheckpointConfiguration": {
        "CheckpointingEnabled": "true",
        "CheckpointInterval": 20000,
        "ConfigurationType": "CUSTOM",
        "MinPauseBetweenCheckpoints": 10000
      }
    }
  }
}

```

Deshabilitar los puntos de control para una nueva aplicación

La siguiente solicitud de ejemplo de la acción [CreateApplication](#) deshabilita los puntos de control al crear una aplicación:

```

{
  "ApplicationName": "MyApplication",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "FlinkApplicationConfiguration": {
        "CheckpointConfiguration": {
          "CheckpointingEnabled": "false"
        }
      }
    }
  }
}

```

```
}  
}
```

Configurar los puntos de control para una aplicación existente

La siguiente solicitud de ejemplo de la acción [UpdateApplication](#) configura los puntos de control para una aplicación existente:

```
{  
  "ApplicationName": "MyApplication",  
  "ApplicationConfigurationUpdate": {  
    "FlinkApplicationConfigurationUpdate": {  
      "CheckpointConfigurationUpdate": {  
        "CheckpointingEnabledUpdate": true,  
        "CheckpointIntervalUpdate": 20000,  
        "ConfigurationTypeUpdate": "CUSTOM",  
        "MinPauseBetweenCheckpointsUpdate": 10000  
      }  
    }  
  }  
}
```

Deshabilitar los puntos de control para una aplicación existente

La siguiente solicitud de ejemplo de la acción [UpdateApplication](#) deshabilita los puntos de control para una aplicación existente:

```
{  
  "ApplicationName": "MyApplication",  
  "ApplicationConfigurationUpdate": {  
    "FlinkApplicationConfigurationUpdate": {  
      "CheckpointConfigurationUpdate": {  
        "CheckpointingEnabledUpdate": false,  
        "CheckpointIntervalUpdate": 20000,  
        "ConfigurationTypeUpdate": "CUSTOM",  
        "MinPauseBetweenCheckpointsUpdate": 10000  
      }  
    }  
  }  
}
```

Administración de copias de seguridad de aplicaciones mediante instantáneas

Una instantánea es la implementación de Managed Service para Apache Flink en un punto de guardado de Apache Flink. Una instantánea es una copia de seguridad del estado de la aplicación activada, creada y gestionada por el usuario o el servicio. Para obtener información sobre los puntos de guardado de Apache Flink, consulte [Puntos de guardado](#) en la [documentación de Apache Flink](#). Con las instantáneas, puede reiniciar una aplicación a partir de una instantánea concreta del estado de la aplicación.

Note

Se recomienda que la aplicación cree una instantánea varias veces al día para que se reinicie correctamente con los datos de estado correctos. La frecuencia correcta de las instantáneas depende de la lógica de negocios de la aplicación. Tomar instantáneas frecuentes le permite recuperar datos más recientes, pero aumenta el costo y requiere más recursos del sistema.

En Managed Service para Apache Flink, las instantáneas se administran mediante las siguientes acciones de la API:

- [CreateApplicationSnapshot](#)
- [DeleteApplicationSnapshot](#)
- [DescribeApplicationSnapshot](#)
- [ListApplicationSnapshots](#)

Para conocer el límite en el número de instantáneas por aplicación, consulte [Cuota](#). Si la aplicación alcanza el límite de instantáneas, la creación manual de una instantánea produce un error con una `LimitExceededException`.

Managed Service para Apache Flink nunca borra las instantáneas. Usted debe eliminar las instantáneas manualmente usando la acción [DeleteApplicationSnapshot](#).

Para cargar una instantánea guardada del estado de la aplicación al iniciar una aplicación, utilice el parámetro [ApplicationRestoreConfiguration](#) de la acción [StartApplication](#) o [UpdateApplication](#).

Este tema contiene las siguientes secciones:

- [Creación de instantáneas automáticas](#)
- [Restauración a partir de una instantánea que contiene datos de estado incompatibles](#)
- [Ejemplos de la API de instantáneas](#)

Creación de instantáneas automáticas

Si `SnapshotsEnabled` se establece en `true` en [ApplicationSnapshotConfiguration](#) para la aplicación, Managed Service para Apache Flink crea y usa instantáneas automáticamente cuando la aplicación se actualiza, escala o detiene para proporcionar una semántica de procesamiento de una sola vez.

Note

Si `ApplicationSnapshotConfiguration::SnapshotsEnabled` se establece en `false`, se perderán datos durante las actualizaciones de la aplicación.

Note

Managed Service para Apache Flink activa puntos de guardado intermedios durante la creación de instantáneas. En la versión 1.15 o superior de Flink, los puntos de guardado intermedios ya no producen efectos secundarios. Consulte [Activación de puntos de guardado](#)

Las instantáneas creadas automáticamente tienen las siguientes cualidades:

- El servicio gestiona la instantánea, pero usted puede verla mediante la acción [ListApplicationSnapshots](#). Las instantáneas creadas automáticamente se tienen en cuenta para el límite de instantáneas.
- Si la aplicación supera el límite de instantáneas, las instantáneas creadas manualmente fallarán, pero el servicio Managed Service para Apache Flink seguirá creando las instantáneas correctamente cuando la aplicación se actualice, escale o detenga. Debe eliminar manualmente las instantáneas mediante la acción [DeleteApplicationSnapshot](#) antes de crear más instantáneas de forma manual.

Restauración a partir de una instantánea que contiene datos de estado incompatibles

Como las instantáneas contienen información sobre los operadores, la restauración de los datos de estado de una instantánea para un operador que ha cambiado desde la versión anterior de la aplicación puede tener resultados inesperados. Una aplicación fallará si intenta restaurar los datos de estado de una instantánea que no corresponde al operador actual. La aplicación con errores se bloqueará en el estado STOPPING o UPDATING.

Para permitir que una aplicación se restaure a partir de una instantánea que contiene datos de estado incompatibles, establezca el parámetro `AllowNonRestoredState` de [FlinkRunConfiguration](#) en `true` mediante la acción [UpdateApplication](#).

Verá el siguiente comportamiento cuando se restaure una aplicación a partir de una instantánea obsoleta:

- Operador agregado: si se agrega un operador nuevo, el punto de guardado no tiene datos de estado para el nuevo operador. No se producirá ningún error y no es necesario establecer `AllowNonRestoredState`.
- Operador eliminado: si se elimina un operador existente, el punto de guardado contiene los datos de estado del operador que falta. Se producirá un error a menos que `AllowNonRestoredState` se establezca en `true`.
- Operador modificado: si se realizan cambios compatibles, como cambiar el tipo de un parámetro por uno compatible, la aplicación puede realizar la restauración a partir de la instantánea obsoleta. Para obtener más información sobre la restauración a partir de instantáneas, consulte [Puntos de guardado](#) en la documentación de Apache Flink. Es posible restaurar una aplicación que utiliza la versión 1.8 o posterior de Apache Flink a partir de una instantánea con un esquema diferente. No se puede restaurar una aplicación que utilice la versión 1.6 de Apache Flink. Para los receptores de confirmación en dos fases, recomendamos utilizar una instantánea del sistema (SwS) en lugar de una instantánea creada por el usuario (`CreateApplicationSnapshot`).

En el caso de Flink, Managed Service para Apache Flink activa puntos de guardado intermedios durante la creación de la instantánea. A partir de la versión 1.15 de Flink, los puntos de guardado intermedios ya no producen efectos secundarios. Consulte [Activación de puntos de guardado](#).

Si necesita reanudar una aplicación que no es compatible con los datos de puntos de guardado existentes, le recomendamos que omita la restauración a partir de la instantánea estableciendo el parámetro `ApplicationRestoreType` de la acción [StartApplication](#) en `SKIP_RESTORE_FROM_SNAPSHOT`.

Para obtener más información sobre cómo Apache Flink trata los datos de estado incompatibles, consulte [Evolución del esquema de estados](#) en la documentación de Apache Flink.

Ejemplos de la API de instantáneas

En esta sección se incluyen solicitudes de ejemplo de acciones de API para usar instantáneas con una aplicación. Para obtener información sobre cómo utilizar un archivo JSON como entrada para una acción de API, consulte [Ejemplo de código de API de Managed Service para Apache Flink](#).

Habilitar las instantáneas para una aplicación

La siguiente solicitud de ejemplo de la acción [UpdateApplication](#) habilita las instantáneas para una aplicación:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationSnapshotConfigurationUpdate": {
      "SnapshotsEnabledUpdate": "true"
    }
  }
}
```

Crear una instantánea

La siguiente solicitud de ejemplo de la acción [CreateApplicationSnapshot](#) crea una instantánea del estado actual de la aplicación:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotName": "MyCustomSnapshot"
}
```

Muestra las instantáneas de una aplicación

La siguiente solicitud de ejemplo de la acción [ListApplicationSnapshots](#) muestra las 50 primeras instantáneas del estado actual de la aplicación:

```
{
```

```
"ApplicationName": "MyApplication",
"Limit": 50
}
```

Mostrar detalles de la instantánea de una aplicación

En la siguiente solicitud de ejemplo de la acción [DescribeApplicationSnapshot](#) se muestran los detalles de una instantánea específica de una aplicación:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotName": "MyCustomSnapshot"
}
```

Eliminar una instantánea

La siguiente solicitud de ejemplo de la acción [DeleteApplicationSnapshot](#) elimina una instantánea guardada anteriormente. Puede obtener el valor de `SnapshotCreationTimestamp` usando [ListApplicationSnapshots](#) o [DeleteApplicationSnapshot](#):

```
{
  "ApplicationName": "MyApplication",
  "SnapshotName": "MyCustomSnapshot",
  "SnapshotCreationTimestamp": 12345678901.0,
}
```

Reinicie una aplicación usando una instantánea con nombre

La siguiente solicitud de ejemplo de la acción [StartApplication](#) inicia la aplicación utilizando el estado guardado de una instantánea específica:

```
{
  "ApplicationName": "MyApplication",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_CUSTOM_SNAPSHOT",
      "SnapshotName": "MyCustomSnapshot"
    }
  }
}
```


Reiniciar una aplicación usando la instantánea más reciente

La siguiente solicitud de ejemplo de la acción [StartApplication](#) inicia la aplicación usando la instantánea más reciente:

```
{
  "ApplicationName": "MyApplication",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

Reiniciar una aplicación sin utilizar ninguna instantánea

La siguiente solicitud de ejemplo de la acción [StartApplication](#) inicia la aplicación sin cargar el estado de la aplicación, incluso si está presente una instantánea:

```
{
  "ApplicationName": "MyApplication",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "SKIP_RESTORE_FROM_SNAPSHOT"
    }
  }
}
```

Reducción horizontal de aplicaciones en Managed Service para Apache Flink

Puede configurar la ejecución paralela de las tareas y la asignación de recursos para que Amazon Managed Service para Apache Flink implemente la reducción horizontal. Para obtener más información sobre cómo Apache Flink programa instancias paralelas de las tareas, consulte [Ejecución paralela](#) en la [documentación de Apache Flink](#).

Temas

- [Configuración del paralelismo de aplicaciones y la KPU ParallelismPer](#)
- [Asignación de unidades de procesamiento de Kinesis](#)

- [Actualización del paralelismo de la aplicación](#)
- [Escalado automático](#)

Configuración del paralelismo de aplicaciones y la KPU ParallelismPer

Para configurar la ejecución paralela de las tareas de la aplicación de Managed Service para Apache Flink (como leer de una fuente o ejecutar un operador), se utilizan las siguientes propiedades [ParallelismConfiguration](#):

- `Parallelism`: utilice esta propiedad para establecer el paralelismo predeterminado de la aplicación de Apache Flink. Todos los operadores, las fuentes y los receptores se ejecutan con este paralelismo, a menos que estén anulados en el código de la aplicación. El valor predeterminado es 1, y el valor máximo es 256.
- `ParallelismPerKPU`: utilice esta propiedad para configurar el número de tareas paralelas que se pueden programar por unidad de procesamiento de Kinesis (KPU) de la aplicación. El valor predeterminado es 1 y el máximo es 8. En el caso de las aplicaciones que tienen operaciones de bloqueo (por ejemplo, E/S), un valor más alto de `ParallelismPerKPU` implica la plena utilización de los recursos de KPU.

Note

El límite de `Parallelism` es igual a `ParallelismPerKPU` veces el límite de las KPU (que tiene un valor predeterminado de 64). El límite de las KPU se puede aumentar solicitando un aumento del límite. Para obtener instrucciones sobre cómo solicitar un aumento de este límite, consulte "Para solicitar un aumento del límite" en [Service Quotas](#).

Para obtener información sobre cómo configurar el paralelismo de tareas para un operador específico, consulte [Configuración del paralelismo: operador](#) en la [documentación de Apache Flink](#).

Asignación de unidades de procesamiento de Kinesis

Provisión de capacidad de Managed Service para Apache Flink como KPU. Una sola KPU le proporciona 1 vCPU y 4 GB de memoria. Por cada KPU asignada, también se proporcionan 50 GB de almacenamiento para aplicaciones en ejecución.

Managed Service para Apache Flink calcula las KPU necesarias para ejecutar la aplicación mediante las propiedades `Parallelism` y `ParallelismPerKPU`, de la siguiente manera:

```
Allocated KPUs for the application = Parallelism/ParallelismPerKPU
```

Managed Service para Apache Flink proporciona rápidamente recursos a las aplicaciones en respuesta a los picos de rendimiento o de la actividad de procesamiento. Elimina los recursos de la aplicación de forma gradual una vez que ha pasado el pico de actividad. Para deshabilitar la asignación automática de recursos, defina el valor `AutoScalingEnabled` en `false`, como se describe más adelante en [Actualización del paralelismo de la aplicación](#).

El límite predeterminado para KPU para su aplicación es de 64. Para obtener instrucciones sobre cómo solicitar un aumento de este límite, consulte "Para solicitar un aumento del límite" en [Service Quotas](#).

Note

Se cobra una KPU adicional por motivos de orquestación. Para obtener más información, consulte [Precios de Managed Service para Apache Flink](#).

Actualización del paralelismo de la aplicación

Esta sección contiene ejemplos de solicitudes de acciones de la API que establecen el paralelismo de una aplicación. Para ver más ejemplos e instrucciones sobre cómo usar los bloques de solicitudes con las acciones de la API, consulte [Ejemplo de código de API de Managed Service para Apache Flink](#).

El siguiente ejemplo de solicitud de la acción [CreateApplication](#) establece el paralelismo al crear una aplicación:

```
{
  "ApplicationName": "string",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
```

```

        "FileKey": "myflink.jar",
        "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
    },
    "CodeContentType": "ZIPFILE"
},
"FlinkApplicationConfiguration": {
    "ParallelismConfiguration": {
        "AutoScalingEnabled": "true",
        "ConfigurationType": "CUSTOM",
        "Parallelism": 4,
        "ParallelismPerKPU": 4
    }
}
}
}
}

```

El siguiente ejemplo de solicitud de la acción [UpdateApplication](#) establece el paralelismo para una aplicación existente:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "ParallelismConfigurationUpdate": {
        "AutoScalingEnabledUpdate": "true",
        "ConfigurationTypeUpdate": "CUSTOM",
        "ParallelismPerKPUUpdate": 4,
        "ParallelismUpdate": 4
      }
    }
  }
}
}
}

```

El siguiente ejemplo de solicitud de la acción [UpdateApplication](#) deshabilita el paralelismo para una aplicación existente:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "ApplicationConfigurationUpdate": {

```

```
"FlinkApplicationConfigurationUpdate": {  
  "ParallelismConfigurationUpdate": {  
    "AutoScalingEnabledUpdate": "false"  
  }  
}  
}
```

Escalado automático

Managed Service para Apache Flink escala elásticamente el paralelismo de la aplicación para adaptarse al rendimiento de datos de su fuente y a la complejidad del operador en la mayoría de los escenarios. Managed Service para Apache Flink supervisa el uso de recursos (CPU) de la aplicación y amplía o reduce el paralelismo de la aplicación elásticamente en consecuencia:

- Su aplicación se amplía (aumenta el paralelismo) si la CloudWatch métrica `containerCPUUtilization` es superior al 75 por ciento o superior durante 15 minutos. Esto significa que la acción `ScaleUp` se activa cuando hay 15 puntos de datos consecutivos con un período de 1 minuto iguales o superiores al 75 por ciento.
- La aplicación se reduce verticalmente (reduce el paralelismo) cuando el uso de la CPU permanece por debajo del 10 por ciento durante seis horas. Esto significa que la acción `ScaleDown` se activa cuando hay 360 puntos de datos consecutivos con un período de 1 minuto iguales o inferiores al 10 por ciento.

Note

Se puede hacer referencia a un período máximo de `containerCPUUtilization` de más de 1 minuto para encontrar la correlación con un punto de datos utilizado para la acción de escalado, pero no es necesario reflejar el momento exacto en que se activa la acción.

Managed Service para Apache Flink no reducirá el valor `CurrentParallelism` de la aplicación a un valor inferior al de la configuración `Parallelism` de la aplicación.

Cuando el servicio de Managed Service para Apache Flink escale su aplicación, aparecerá en estado `AUTOSCALING`. Puede comprobar el estado actual de su solicitud mediante las [DescribeApplication](#) acciones o [ListApplications](#). Mientras el servicio escala tu aplicación, la única acción válida de la API que puedes usar es [StopApplication](#) establecer el `Force` parámetro en `true`.

Puede usar la propiedad `AutoScalingEnabled` (parte de [FlinkApplicationConfiguration](#)) para habilitar o deshabilitar el comportamiento de escalado automático. Se le cobrarán a su cuenta AWS las KPU suministradas por Managed Service para Apache Flink, lo cual es una función de la configuración `parallelism` y `parallelismPerKPU` de la aplicación. Si se produce un pico de actividad, ello aumentará los costos de Managed Service para Apache Flink.

Para obtener más información sobre precios, consulte [Precios de Amazon Managed Service para Apache Flink](#).

Tenga en cuenta lo siguiente en relación con el escalado de la aplicación:

- El escalado automático está habilitado de forma predeterminada.
- El escalado no se aplica a los cuadernos de Studio. Sin embargo, si implementa un cuaderno de Studio como una aplicación de estado perdurable, el escalado se aplicará a la aplicación implementada.
- La aplicación tiene un límite predeterminado de 64 KPU. Para obtener más información, consulte [Cuota](#).
- Cuando el escalado automático actualiza el paralelismo de la aplicación, la aplicación sufre un tiempo de inactividad. Para evitar este tiempo de inactividad, haga lo siguiente:
 - Deshabilite el escalado automático
 - Configura la `parallelism` y `parallelismPerKPU` con la [UpdateApplication](#) acción de tu aplicación. Para obtener más información sobre cómo realizar la configuración de paralelismo de la aplicación, consulte [the section called “Actualización del paralelismo de la aplicación”](#).
 - Supervise periódicamente el uso de los recursos de la aplicación para comprobar que la aplicación tenga una configuración de paralelismo correcta adecuada para su carga de trabajo. Para obtener información sobre supervisión del uso de recursos de asignación, consulte [the section called “Métricas y dimensiones en Managed Service para Apache Flink”](#).

Consideraciones sobre `maxParallelism`

- La lógica del escalado automático evitará escalar una tarea de Flink a un paralelismo que provoque interferencias con la tarea y el `maxParallelism` del operador. Por ejemplo, en caso de un trabajo simple con solo un origen y un receptor, donde el origen tiene un `maxParallelism` de 16 y el sink tiene 8, no realizaremos el escalado automático del trabajo a valores superiores a 8.

- Si no se configuró `maxParallelism` para un trabajo, Flink usará 128 por defecto. Por lo tanto, si considera que un trabajo debe ejecutarse con un paralelismo superior a 128, tendrá que configurar ese número para la aplicación.
- Si espera que se produzca el escalado automático del trabajo, pero no ve que ocurra, asegúrese de que los valores de `maxParallelism` lo permitan.

Para obtener información adicional, consulte [Supervisión y escalado automático mejorados para Apache Flink](#)

Para ver un ejemplo, consulte [kda-flink-app-autoscaling](#).

Uso del etiquetado

En esta sección, se describe cómo añadir etiquetas de metadatos de clave-valor a las aplicaciones de Managed Service para Apache Flink. Estas etiquetas se pueden utilizar para lo siguiente:

- Determinar la facturación para aplicaciones individuales de Managed Service para Apache Flink. Para obtener más información, consulte [Uso de etiquetas de asignación de costos](#) en la Guía de administración de costos y facturación.
- Controlar el acceso a los recursos de la aplicación en función de las etiquetas. Para obtener más información, consulte [Control de acceso mediante etiquetas](#) en la Guía del usuario de AWS Identity and Access Management.
- Fines definidos por el usuario. Puede definir la funcionalidad de la aplicación en función de la presencia de etiquetas del usuario.

Tenga en cuenta la siguiente información sobre el etiquetado:

- El número máximo de etiquetas de la aplicación incluye las etiquetas del sistema. El número máximo de etiquetas de la aplicación definidas por el usuario es 50.
- Si una acción incluye una lista de etiquetas que tiene valores Key duplicados, el servicio genera una `InvalidArgumentException`.

Este tema contiene las siguientes secciones:

- [Adición de etiquetas al crear una aplicación](#)
- [Adición o actualización de etiquetas para una aplicación existente](#)

- [Mostrar las etiquetas de una aplicación](#)
- [Eliminación de etiquetas de una aplicación](#)

Adición de etiquetas al crear una aplicación

Puede añadir etiquetas al crear una aplicación utilizando el parámetro `tags` de la acción [CreateApplication](#).

En la siguiente solicitud de ejemplo, se muestra el nodo `Tags` de una solicitud `CreateApplication`:

```
"Tags": [  
  {  
    "Key": "Key1",  
    "Value": "Value1"  
  },  
  {  
    "Key": "Key2",  
    "Value": "Value2"  
  }  
]
```

Adición o actualización de etiquetas para una aplicación existente

Puede añadir etiquetas a una aplicación utilizando la acción [TagResource](#). No se pueden agregar etiquetas a una aplicación utilizando la acción [UpdateApplication](#).

Para actualizar una etiqueta existente, añada otra etiqueta con la misma clave.

En la siguiente solicitud de ejemplo de la acción `TagResource`, se añaden nuevas etiquetas o se actualizan las etiquetas existentes:

```
{  
  "ResourceARN": "string",  
  "Tags": [  
    {  
      "Key": "NewTagKey",  
      "Value": "NewTagValue"  
    },  
    {
```



```
        "Key": "ExistingKeyOfTagToUpdate",
        "Value": "NewValueForExistingTag"
    }
  ]
}
```

Mostrar las etiquetas de una aplicación

Para obtener una lista de las etiquetas existentes, utilice la acción [ListTagsForResource](#).

En la siguiente solicitud de ejemplo de la acción `ListTagsForResource`, se muestran las etiquetas de una aplicación:

```
{
  "ResourceARN": "arn:aws:kinesisanalyticsus-west-2:012345678901:application/MyApplication"
}
```

Eliminación de etiquetas de una aplicación

Para eliminar las etiquetas de una aplicación, use la acción [UntagResource](#).

En la siguiente solicitud de ejemplo de la acción `UntagResource`, se eliminan etiquetas de una aplicación:

```
{
  "ResourceARN": "arn:aws:kinesisanalyticsus-west-2:012345678901:application/MyApplication",
  "TagKeys": [ "KeyOfFirstTagToRemove", "KeyOfSecondTagToRemove" ]
}
```

Uso de CloudFormation con Managed Service para Apache Flink

Los siguientes ejercicios muestran cómo iniciar una aplicación de Flink creada mediante AWS CloudFormation usando una función de Lambda en la misma pila.

Antes de empezar

Antes de comenzar este ejercicio, siga los pasos para crear una aplicación Flink usando AWS CloudFormation en [AWS::KinesisAnalytics::Application](#).

Escritura de una función de Lambda

Para iniciar una aplicación de Flink después de crearla o actualizarla, utilizamos la API `kinesisanalyticsv2` [start-application](#). La llamada se activará mediante un evento AWS CloudFormation después de la creación de la aplicación Flink. Explicaremos cómo configurar la pila para activar la función de Lambda más adelante en este ejercicio, pero primero nos centraremos en la declaración de la función de Lambda y su código. En este ejemplo, utilizamos el tiempo de ejecución de Python3.8.

```
StartApplicationLambda:
  Type: AWS::Lambda::Function
  DependsOn: StartApplicationLambdaRole
  Properties:
    Description: Starts an application when invoked.
    Runtime: python3.8
    Role: !GetAtt StartApplicationLambdaRole.Arn
    Handler: index.lambda_handler
    Timeout: 30
    Code:
      ZipFile: |
        import logging
        import cfnresponse
        import boto3

        logger = logging.getLogger()
        logger.setLevel(logging.INFO)

        def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
                application_name = event['ResourceProperties']['ApplicationName']

                # filter out events other than Create or Update,
                # you can also omit Update in order to start an application on Create
                # only.
                if event['RequestType'] not in ["Create", "Update"]:
                    logger.info('No-op for Application {} because CFN RequestType {} is
                    filtered'.format(application_name, event['RequestType']))
                    cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

            return
```

```
# use kinesisanalyticsv2 API to start an application.
client_kda = boto3.client('kinesisanalyticsv2',
region_name=event['ResourceProperties']['Region'])

# get application status.
describe_response =
client_kda.describe_application(ApplicationName=application_name)
application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

# an application can be started from 'READY' status only.
if application_status != 'READY':
    logger.info('No-op for Application {} because ApplicationStatus {} is
filtered'.format(application_name, application_status))
    cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

return

# create RunConfiguration.
run_configuration = {
    'ApplicationRestoreConfiguration': {
        'ApplicationRestoreType': 'RESTORE_FROM_LATEST_SNAPSHOT',
    }
}

logger.info('RunConfiguration for Application {}:
{}'.format(application_name, run_configuration))

# this call doesn't wait for an application to transfer to 'RUNNING'
state.
client_kda.start_application(ApplicationName=application_name,
RunConfiguration=run_configuration)

logger.info('Started Application: {}'.format(application_name))
cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
except Exception as err:
    logger.error(err)
    cfnresponse.send(event, context, cfnresponse.FAILED, {"Data": str(err)})
```

En el código anterior, Lambda procesará los eventos de AWS CloudFormation entrantes, filtrará todo excepto Create y Update, obtendrá el estado de la aplicación y la iniciará si el estado es

READY. Para obtener el estado de la aplicación, debe crear el rol de Lambda, como se muestra a continuación:

Creación de rol de Lambda

Deberá crear un rol para que Lambda “hable” correctamente con la aplicación y escriba registros. Este rol usará políticas administradas predeterminadas, pero es posible que desee restringirlo usando políticas personalizadas.

```
StartApplicationLambdaRole:
  Type: AWS::IAM::Role
  DependsOn: TestFlinkApplication
  Properties:
    Description: A role for lambda to use while interacting with an application.
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - lambda.amazonaws.com
          Action:
            - sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
      - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
    Path: /
```

Tenga en cuenta que los recursos de Lambda se crearán después de crear la aplicación Flink en la misma pila, ya que dependen de ella.

Invocación de la función de Lambda

Ahora solo queda invocar la función de Lambda. Esto se hace mediante un [recurso personalizado](#).

```
StartApplicationLambdaInvoke:
  Description: Invokes StartApplicationLambda to start an application.
  Type: AWS::CloudFormation::CustomResource
  DependsOn: StartApplicationLambda
  Version: "1.0"
  Properties:
    ServiceToken: !GetAtt StartApplicationLambda.Arn
```

```
Region: !Ref AWS::Region
ApplicationName: !Ref TestFlinkApplication
```

Esto es todo lo que necesita para iniciar la aplicación Flink con Lambda. Ahora puede crear su propia pila o utilizar el ejemplo completo que aparece a continuación para ver cómo funcionan todos esos pasos en la práctica.

Ejemplo completo

El siguiente ejemplo es una versión ligeramente ampliada de los pasos anteriores con un ajuste `RunConfiguration` adicional realizado mediante los [parámetros de la plantilla](#). Esta es una pila funcional para que la pruebe. Asegúrese de leer las notas adjuntas:

stack.yaml

```
Description: 'kinesisanalyticsv2 CloudFormation Test Application'
Parameters:
  ApplicationRestoreType:
    Description: ApplicationRestoreConfiguration option, can
    be SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT or
    RESTORE_FROM_CUSTOM_SNAPSHOT.
    Type: String
    Default: SKIP_RESTORE_FROM_SNAPSHOT
    AllowedValues: [ SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT,
    RESTORE_FROM_CUSTOM_SNAPSHOT ]
  SnapshotName:
    Description: ApplicationRestoreConfiguration option, name of a snapshot to restore
    to, used with RESTORE_FROM_CUSTOM_SNAPSHOT ApplicationRestoreType.
    Type: String
    Default: ''
  AllowNonRestoredState:
    Description: FlinkRunConfiguration option, can be true or false.
    Default: true
    Type: String
    AllowedValues: [ true, false ]
  CodeContentBucketArn:
    Description: ARN of a bucket with application code.
    Type: String
  CodeContentFileKey:
    Description: A jar filename with an application code inside a bucket.
    Type: String
Conditions:
  IsSnapshotNameEmpty: !Equals [ !Ref SnapshotName, '' ]
```

```
Resources:
  TestServiceExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - kinesisanalytics.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisFullAccess
        - arn:aws:iam::aws:policy/AmazonS3FullAccess
      Path: /
  InputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  OutputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  TestFlinkApplication:
    Type: 'AWS::kinesisanalyticsv2::Application'
    Properties:
      ApplicationName: 'CFNTestFlinkApplication'
      ApplicationDescription: 'Test Flink Application'
      RuntimeEnvironment: 'FLINK-1_15'
      ServiceExecutionRole: !GetAtt TestServiceExecutionRole.Arn
      ApplicationConfiguration:
        EnvironmentProperties:
          PropertyGroups:
            - PropertyGroupId: 'KinesisStreams'
          PropertyMap:
            INPUT_STREAM_NAME: !Ref InputKinesisStream
            OUTPUT_STREAM_NAME: !Ref OutputKinesisStream
            AWS_REGION: !Ref AWS::Region
      FlinkApplicationConfiguration:
        CheckpointConfiguration:
          ConfigurationType: 'CUSTOM'
          CheckpointingEnabled: True
          CheckpointInterval: 1500
```

```
    MinPauseBetweenCheckpoints: 500
  MonitoringConfiguration:
    ConfigurationType: 'CUSTOM'
    MetricsLevel: 'APPLICATION'
    LogLevel: 'INFO'
  ParallelismConfiguration:
    ConfigurationType: 'CUSTOM'
    Parallelism: 1
    ParallelismPerKPU: 1
    AutoScalingEnabled: True
  ApplicationSnapshotConfiguration:
    SnapshotsEnabled: True
  ApplicationCodeConfiguration:
    CodeContent:
      S3ContentLocation:
        BucketARN: !Ref CodeContentBucketArn
        FileKey: !Ref CodeContentFileKey
      CodeContentType: 'ZIPFILE'
StartApplicationLambdaRole:
  Type: AWS::IAM::Role
  DependsOn: TestFlinkApplication
  Properties:
    Description: A role for lambda to use while interacting with an application.
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - lambda.amazonaws.com
          Action:
            - sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
      - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
    Path: /
StartApplicationLambda:
  Type: AWS::Lambda::Function
  DependsOn: StartApplicationLambdaRole
  Properties:
    Description: Starts an application when invoked.
    Runtime: python3.8
    Role: !GetAtt StartApplicationLambdaRole.Arn
    Handler: index.lambda_handler
```

```
Timeout: 30
Code:
ZipFile: |
    import logging
    import cfnresponse
    import boto3

    logger = logging.getLogger()
    logger.setLevel(logging.INFO)

    def lambda_handler(event, context):
        logger.info('Incoming CFN event {}'.format(event))

        try:
            application_name = event['ResourceProperties']['ApplicationName']

            # filter out events other than Create or Update,
            # you can also omit Update in order to start an application on Create
            only.

            if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
                filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

            return

            # use kinesisanalyticsv2 API to start an application.
            client_kda = boto3.client('kinesisanalyticsv2',
            region_name=event['ResourceProperties']['Region'])

            # get application status.
            describe_response =
            client_kda.describe_application(ApplicationName=application_name)
            application_status = describe_response['ApplicationDetail']
            ['ApplicationStatus']

            # an application can be started from 'READY' status only.
            if application_status != 'READY':
                logger.info('No-op for Application {} because ApplicationStatus {} is
                filtered'.format(application_name, application_status))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

            return
```



```

        # create RunConfiguration from passed parameters.
        run_configuration = {
            'FlinkRunConfiguration': {
                'AllowNonRestoredState': event['ResourceProperties']
['AllowNonRestoredState'] == 'true'
            },
            'ApplicationRestoreConfiguration': {
                'ApplicationRestoreType': event['ResourceProperties']
['ApplicationRestoreType'],
            }
        }

        # add SnapshotName to RunConfiguration if specified.
        if event['ResourceProperties']['SnapshotName'] != '':
            run_configuration['ApplicationRestoreConfiguration']['SnapshotName'] =
event['ResourceProperties']['SnapshotName']

        logger.info('RunConfiguration for Application {}:
{}'.format(application_name, run_configuration))

        # this call doesn't wait for an application to transfer to 'RUNNING'
state.
        client_kda.start_application(ApplicationName=application_name,
RunConfiguration=run_configuration)

        logger.info('Started Application: {}'.format(application_name))
        cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
    except Exception as err:
        logger.error(err)
        cfnresponse.send(event, context, cfnresponse.FAILED, {"Data": str(err)})
StartApplicationLambdaInvoke:
Description: Invokes StartApplicationLambda to start an application.
Type: AWS::CloudFormation::CustomResource
DependsOn: StartApplicationLambda
Version: "1.0"
Properties:
    ServiceToken: !GetAtt StartApplicationLambda.Arn
    Region: !Ref AWS::Region
    ApplicationName: !Ref TestFlinkApplication
    ApplicationRestoreType: !Ref ApplicationRestoreType
    SnapshotName: !Ref SnapshotName
    AllowNonRestoredState: !Ref AllowNonRestoredState

```

Una vez más, es posible que desee ajustar los roles de Lambda y de la propia aplicación.

Antes de crear la pila anterior, no olvide especificar los parámetros.

parameters.json

```
[
  {
    "ParameterKey": "CodeContentBucketArn",
    "ParameterValue": "YOUR_BUCKET_ARN"
  },
  {
    "ParameterKey": "CodeContentFileKey",
    "ParameterValue": "YOUR_JAR"
  },
  {
    "ParameterKey": "ApplicationRestoreType",
    "ParameterValue": "SKIP_RESTORE_FROM_SNAPSHOT"
  },
  {
    "ParameterKey": "AllowNonRestoredState",
    "ParameterValue": "true"
  }
]
```

Reemplace YOUR_BUCKET_ARN y YOUR_JAR con sus requerimientos específicos. Puede seguir esta [guía](#) para crear un bucket de Amazon S3 y un jar de aplicación.

Ahora cree la pila (sustituya YOUR_REGION por la región que prefiera, por ejemplo, us-east-1):

```
aws cloudformation create-stack --region YOUR_REGION --template-body "file://
stack.yaml" --parameters "file://parameters.json" --stack-name "TestManaged Service for
Apache FlinkStack" --capabilities CAPABILITY_NAMED_IAM
```

Ahora puede ir a <https://console.aws.amazon.com/cloudformation> y ver el progreso. Una vez creada, debería ver su aplicación de Flink en el estado Starting. Puede que tarde algunos minutos en comenzar a Running.

Para obtener más información, consulte lo siguiente:

- [Cuatro formas de recuperar cualquier propiedad de servicio de AWS mediante AWS CloudFormation \(parte 1 de 3\)](#).

- [Tutorial: Búsqueda de ID de imagen de Amazon Machine.](#)

Uso del panel de control de Apache Flink con Managed Service para Apache Flink

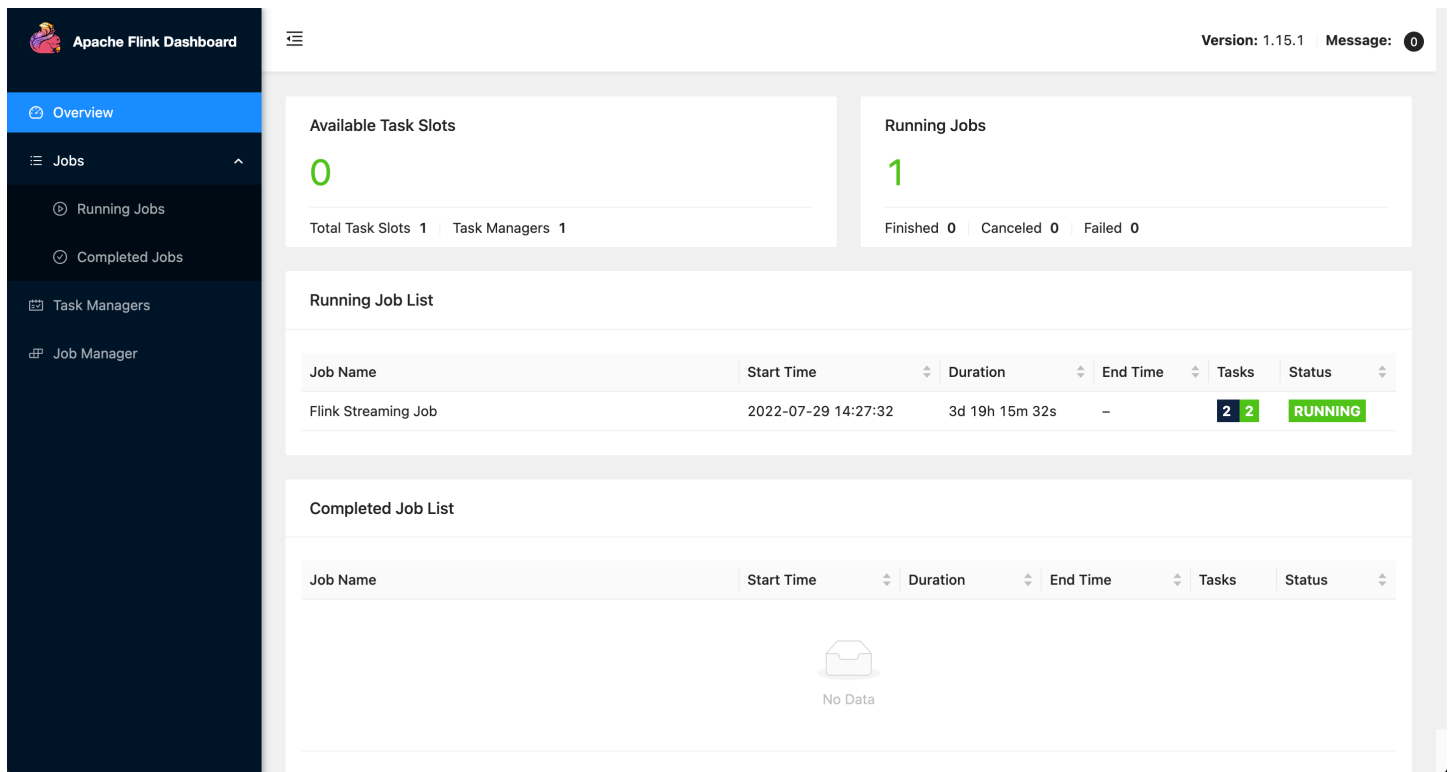
Puede utilizar el panel de control Apache Flink de su aplicación para supervisar el estado de la aplicación Managed Service para Apache Flink. El panel de control de su aplicación muestra la siguiente información:

- Recursos en uso, incluidos los administradores de tareas y los espacios de tareas.
- Información sobre los trabajos, incluidos los que están en ejecución, los que se han completado, los cancelados y los que han fallado.

Para obtener información sobre los administradores de tareas, los espacios de tareas y los trabajos de Apache Flink, consulte la [arquitectura de Apache Flink](#) en el sitio web de Apache Flink.

Note lo siguiente sobre el uso del panel de control de Apache Flink con Managed Service para Apache Flink

- El panel de control de Apache Flink para Managed Service para Apache Flink es de solo lectura. No puede realizar cambios a Managed Service para Apache Flink usando el panel de control de Apache Flink.
- El panel de control de Apache Flink no es compatible con Microsoft Internet Explorer.



The screenshot displays the Apache Flink Dashboard interface. On the left is a dark sidebar with navigation options: Overview (selected), Jobs, Running Jobs, Completed Jobs, Task Managers, and Job Manager. The main content area shows a top navigation bar with 'Version: 1.15.1' and 'Message: 0'. Below this, there are two summary cards: 'Available Task Slots' showing '0' and 'Running Jobs' showing '1'. The 'Available Task Slots' card also lists 'Total Task Slots 1' and 'Task Managers 1'. The 'Running Jobs' card lists 'Finished 0', 'Canceled 0', and 'Failed 0'. Below these is a 'Running Job List' table with one entry: 'Flink Streaming Job' with a start time of '2022-07-29 14:27:32', a duration of '3d 19h 15m 32s', and a status of 'RUNNING' with 2 tasks. At the bottom, there is a 'Completed Job List' section which is currently empty, showing 'No Data'.

Acceder al panel de control de Apache Flink de su aplicación

Puede acceder al panel de control de Apache Flink de su aplicación a través de la consola de Managed Service para Apache Flink o solicitando un punto de conexión de URL seguro mediante la CLI.

Acceder a su panel de control de Apache Flink usando la consola de Managed Service para Apache Flink.

Para acceder al panel de control de Apache Flink de su aplicación desde la consola, seleccione el panel de control de Apache Flink en la página de su aplicación.

Note

Al abrir el panel desde la consola de Managed Service para Apache Flink, la URL que genere la consola será válida durante 12 horas.

Acceder a su panel de control de Apache Flink usando la CLI de Managed Service para Apache Flink

Puede utilizar la CLI de Managed Service para Apache Flink para generar una URL que le permita acceder al panel de control de la aplicación. La URL que genere es válida durante un período especificado.

Note

Si no accede a la URL generada en tres minutos, dejará de ser válida.

Para generar la URL de su panel de control, utilice la acción [CreateApplicationPreSignedURL](#). Puede especificar los siguientes valores para la acción:

- El nombre de la aplicación.
- El tiempo en segundos durante el que la URL será válida
- Se especifica FLINK_DASHBOARD_URL como tipo de URL.

Versiones de lanzamiento

Este tema contiene información sobre las características compatibles y las versiones de los componentes recomendadas para cada versión de Managed Service para Apache Flink.

Amazon Managed Service para Apache Flink versión 1.15.2

Managed Service para Apache Flink admite las siguientes características nuevas en Apache 1.15.2

Característica	Descripción	Referencia de Apache FLIP
Async Sink	Un AWS marco colaborativo para crear destinos asíncronos que permite a los desarrolladores crear AWS conectores personalizados con menos de la mitad del esfuerzo anterior. Para obtener más información, consulte The Generic Asynchronous Base Sink .	FLIP-171: Async Sink .
Receptor de Kinesis Data Firehose	AWS ha creado un nuevo receptor Amazon Kinesis Firehose que utiliza el marco Async.	Receptor de Amazon Kinesis Data Firehose
Detener con punto de control	Detener con punto de control garantiza una operación de detención limpia y, lo que es más importante, respalda la semántica exactamente una vez para los clientes que confían en ella.	FLIP-34: Finalizar/suspender un trabajo con Savepoint .
Desacoplamiento de Scala	Los usuarios ahora pueden aprovechar la API de Java desde cualquier versión de	FLIP-28: Objetivo a largo plazo de hacer que Flink-Tab le funcione sin Scala .

Característica	Descripción	Referencia de Apache FLIP
	Scala, incluida Scala 3. Los clientes deberán incluir la biblioteca estándar de Scala que elijan en sus aplicaciones de Scala.	
Scala	Véase el desacoplamiento de Scala más arriba	FLIP-28: Objetivo a largo plazo de hacer que Flink-Tab le funcione sin Scala.
Métricas de conectores unificados	Flink ha definido métricas estándar para trabajos, tareas y operadores. Managed Service para Apache Flink seguirá siendo compatible con las métricas de receptor y origen y, en la versión 1.15, se introducirá <code>numRestarts</code> en paralelo con <code>fullRestarts</code> para las Métricas de disponibilidad.	FLIP-33: Estandarizar las métricas de los conectores y FLIP-179: Exponer las métricas estandarizadas de los operadores.
Creación de un punto de control de tareas terminadas	Esta característica está habilitada de forma predeterminada en Flink 1.15 y permite seguir realizando puntos de control incluso si algunas partes del gráfico del trabajo han terminado de procesar todos los datos, lo que podría ocurrir si contiene fuentes limitadas (por lotes).	FLIP-147: Puntos de control de soporte una vez finalizadas las tareas.

Cambios en Amazon Managed Service para Apache Flink con Apache Flink 1.15

Bloc de notas de Studio

Managed Service para Apache Flink Studio ahora es compatible con Apache Flink 1.15. Managed Service para Apache Flink Studio utiliza los blocs de notas de Apache Zeppelin para ofrecer una experiencia de desarrollo de interfaz única para desarrollar, depurar código y ejecutar aplicaciones de procesamiento de flujos de Apache Flink. Puede obtener más información sobre Managed Service para Apache Flink Studio y cómo empezar en [Uso de un bloc de notas de Studio con Managed Service para Apache Flink](#).

Conector EFO

Al actualizar a la versión 1.15 de Managed Service para Apache Flink, asegúrese de utilizar el conector EFO más reciente, es decir, cualquier versión 1.15.3 o posterior. Para obtener más información sobre el motivo, consulte [FLINK-29324](#).

Desacoplamiento de Scala

Comenzando con Flink 1.15.2, deberá incluir la biblioteca estándar de Scala que elija en sus aplicaciones de Scala.

Receptor de Kinesis Data Firehose

Al actualizar a la versión 1.15 de Managed Service para Apache Flink, asegúrese de utilizar la versión más reciente del receptor de [Amazon Kinesis Data Firehose](#).

Conectores Kafka

Al actualizar a Amazon Managed Service para Apache Flink para la versión 1.15 de Apache Flink, asegúrese de utilizar la versión más reciente de las API del conector Kafka. Apache Flink ha dejado de utilizar [FlinkKafkaConsumer](#) y [FlinkKafkaProducer](#). Estas API del receptor de Kafka no se pueden asignar a Kafka para Flink 1.15. Asegúrese de utilizar [KafkaSource](#) y [KafkaSink](#).

Componentes

Componente	Versión
Java	11 (recomendado)

Componente	Versión
Scala	2.12
Managed Service para Apache Flink para el tiempo de ejecución de Flink (aws-kinesisanalytics-runtime)	1.2.0
Conector AWS Kinesis (flink-connector-kinesis)	1.15.4
Apache Beam (solo aplicaciones Beam)	2.33.0, con Jackson versión 2.12.2

Uso de un bloc de notas de Studio con Managed Service para Apache Flink

Los blocs de notas de Studio para Managed Service para Apache Flink le permiten consultar flujos de datos de forma interactiva en tiempo real y crear y ejecutar fácilmente aplicaciones de procesamiento de flujos mediante SQL, Python y Scala estándares. Con unos pocos clics en la consola de Administración de AWS, puede iniciar un bloc de notas sin servidor para consultar flujos de datos y obtener resultados en cuestión de segundos.

Un bloc de notas es un entorno de desarrollo basado en la web. Con los blocs de notas, obtiene una experiencia de desarrollo interactiva sencilla combinada con las capacidades avanzadas que proporciona Apache Flink. Los blocs de notas de Studio utilizan blocs de notas con tecnología [Apache Zeppelin](#) y [Apache Flink](#) como motor de procesamiento de flujos. Los blocs de notas de Studio combinan estas tecnologías a la perfección para que los desarrolladores de todo nivel de habilidades puedan acceder a los análisis avanzados de los flujos de datos.

Apache Zeppelin proporciona a sus blocs de notas de Studio un conjunto completo de herramientas de análisis, entre las que se incluyen las siguientes:

- Visualización de datos
- Exportación de datos a archivos
- Control del formato de salida para facilitar el análisis

Para empezar a utilizar Managed Service para Apache Flink y Apache Zeppelin, consulte [Tutorial sobre cómo crear un bloc de notas de Studio](#). Para obtener más información acerca de Apache Zeppelin, consulte la [documentación de Apache Zeppelin](#).

Con un cuaderno, puede modelar consultas utilizando la [API y SQL de Apache Flink Table](#) en SQL, Python o Scala, o la [DataStream API](#) en Scala. Con unos pocos clics, puede convertir el bloc de notas de Studio en una aplicación de procesamiento de flujos de Managed Service para Apache Flink que se ejecute de forma continua y no interactiva para sus cargas de trabajo de producción.

Este tema contiene las siguientes secciones:

- [Crear un bloc de notas de Studio](#)
- [Análisis interactivo de los datos de streaming](#)

- [Implementación como una aplicación con un estado duradero](#)
- [Permisos de IAM para blocs de notas de Studio](#)
- [Conectores y dependencias](#)
- [Funciones definidas por el usuario](#)
- [Activación de puntos de control](#)
- [Trabajar con AWS Glue](#)
- [Ejemplos y tutoriales](#)
- [Solución de problemas](#)
- [Apéndice: Creación de políticas de IAM personalizadas](#)

Crear un bloc de notas de Studio

Un bloc de notas de Studio contiene consultas o programas escritos en SQL, Python o Scala que se ejecutan en datos de streaming y devuelven resultados analíticos. La aplicación se crea mediante la consola o la CLI y se proporcionan consultas para analizar los datos del origen de datos.

Su aplicación tiene los siguientes componentes:

- Un origen de datos, como un clúster de Amazon MSK, un flujo de datos de Kinesis o un bucket de Amazon S3.
- Una base de datos de AWS Glue. Esta base de datos contiene tablas en las que se almacenan el origen de datos, los esquemas de destino y los puntos de conexión. Para obtener más información, consulte [Trabajar con AWS Glue](#).
- Verifique el código de la aplicación. El código implementa la consulta o el programa de análisis.
- La configuración de la aplicación y las propiedades del tiempo de ejecución. Para obtener información acerca de la configuración de la aplicación y las propiedades del tiempo de ejecución, consulte los siguientes temas en la [Guía para desarrolladores de aplicaciones Apache Flink](#):
 - Paralelismo y escalado de la aplicación: utilice la configuración de paralelismo de la aplicación para controlar el número de consultas que la aplicación puede ejecutar simultáneamente. Las consultas también pueden aprovechar el aumento del paralelismo si tienen varias rutas de ejecución, por ejemplo, en las siguientes circunstancias:
 - Al procesar varios fragmentos de un flujo de datos de Kinesis
 - Al particionar los datos mediante el operador KeyBy.

- Cuando se utilizan operadores de ventanas múltiples

Para obtener más información acerca del escalado de aplicaciones, consulte [Escalado de aplicaciones en Managed Service para Apache Flink](#).

- Registro y monitorización: para obtener información sobre el registro y la monitorización de aplicaciones, consulte [Registro y monitorización en Amazon Managed Service para Apache Flink](#).
- La aplicación utiliza puntos de control y puntos de almacenamiento para garantizar la tolerancia a los errores. Los puntos de control y los puntos de almacenamiento no están habilitados de forma predeterminada en los blocs de notas de Studio.

Puede crear su bloc de notas de Studio utilizando AWS Management Console o AWS CLI.

Al crear la aplicación desde la consola, tiene las siguientes opciones:

- En la consola de Amazon MSK, elija su clúster y, a continuación, elija Procesar datos en tiempo real.
- En la consola de Kinesis Data Streams, elija su flujo de datos y, en la pestaña Aplicaciones, elija Procesar datos en tiempo real.
- En la consola de Managed Service para Apache Flink, seleccione la pestaña Studio y, a continuación, seleccione Crear bloc de notas de Studio.

Para ver un tutorial, consulte [Detección de eventos con servicio administrado para Apache Flink](#).

Para ver un ejemplo de una solución de bloc de notas de Studio más avanzada, consulta [Apache Flink en Amazon Managed Service para Apache Flink Studio](#).

Análisis interactivo de los datos de streaming

Utiliza un portátil sin servidor con tecnología Apache Zeppelin para interactuar con sus datos de streaming. Su bloc de notas puede tener varias notas y cada nota puede tener uno o más párrafos en los que puede escribir el código.

El siguiente ejemplo de consulta SQL muestra cómo recuperar datos de un origen de datos:

```
%flink.ssql(type=update)
```

```
select * from stock;
```

Para ver más ejemplos de consultas SQL de Flink Streaming, consulte [Ejemplos y tutoriales](#) a continuación y [Consultas](#) en la [documentación de Apache Flink](#).

Puede utilizar las consultas SQL de Flink del bloc de notas de Studio para consultar los datos de streaming. También puede usar Python (API de tabla) y Scala (API de tabla y de flujo de datos) para escribir programas que consulten sus datos de streaming de forma interactiva. Puede ver los resultados de sus consultas o programas, actualizarlos en cuestión de segundos y volver a ejecutarlos para ver los resultados actualizados.

Intérpretes de Flink

Usted especifica el idioma que utiliza Managed Service para Apache Flink para ejecutar su aplicación mediante un intérprete. Puede usar los siguientes intérpretes con Managed Service para Apache Flink:

Name	Clase	Descripción
%flink	FlinkInterpreter	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment
%flink.pyflink	PyFlinkInterpreter	Provides a python environment
%flink.ipynk	IPyFlinkInterpreter	Provides an ipython environment
%flink.ssql	FlinkStreamSqlInterpreter	Provides a stream sql environment
%flink.bsql	FlinkBatchSqlInterpreter	Provides a batch sql environment

Para obtener más información sobre los intérpretes de Flink, consulte [Intérprete de Flink para Apache Zeppelin](#).

Si utiliza `%flink.pyflink` o `%flink.ipyflink` como intérpretes, necesitará usar `ZeppelinContext` para visualizar los resultados en el bloc de notas.

Para ver ejemplos más PyFlink específicos, consulte [Consulte sus flujos de datos de forma interactiva mediante Managed Service para Apache Flink Studio y Python.](#)

Variables de entorno de la tabla de Apache Flink

Apache Zeppelin proporciona acceso a los recursos del entorno de la tabla mediante variables de entorno.

Se accede a los recursos del entorno de tablas de Scala con las siguientes variables:

Variable	Recurso
<code>senv</code>	<code>StreamExecutionEnvironment</code>
<code>stenv</code>	<code>StreamTableEnvironment</code> para el planificador de parpadeos

Puede acceder a los recursos del entorno de tablas de Python con las siguientes variables:

Variable	Recurso
<code>s_env</code>	<code>StreamExecutionEnvironment</code>
<code>st_env</code>	<code>StreamTableEnvironment</code> para Blink Planner

Para obtener más información sobre el uso de entornos de tablas, consulte [Crear una TableEnvironment](#) en la documentación de [Apache Flink](#).

Implementación como una aplicación con un estado duradero

Puede crear el código y exportarlo a Amazon S3. Puede convertir el código que escribió en su nota en una aplicación de procesamiento de flujo en funcionamiento continuo. Existen dos modos de ejecutar una aplicación de Apache Flink en Managed Service para Apache Flink: usando un bloc de notas de Studio, puede desarrollar código de forma interactiva, ver los resultados del código en

tiempo real y visualizarlos en la nota. Después de implementar una nota para que se ejecute en modo streaming, Managed Service para Apache Flink crea una aplicación para usted que se ejecuta de forma continua, lee los datos de sus fuentes, escribe en sus destinos, mantiene el estado de la aplicación de larga duración y se escala automáticamente en función del rendimiento de los flujos de origen.

Note

El bucket de S3 al que exporte el código de la aplicación debe estar en la misma región que el bloc de notas de Studio.

Solo puedes implementar una nota desde tu bloc de notas de Studio si cumple los siguientes criterios:

- Los párrafos deben ordenarse secuencialmente. Al implementar la aplicación, todos los párrafos de una nota se ejecutarán secuencialmente (left-to-right, top-to-bottom) tal como aparecen en la nota. Para comprobar este orden, selecciona Ejecutar todos los párrafos en la nota.
- Su código es una combinación de Python y SQL o Scala y SQL. No admitimos Python y Scala juntos en este momento. `deploy-as-application`
- Su nota solo debe tener los siguientes intérpretes: `%flink`, `%flink.ssql`, `%flink.pyflink`, `%flink.ipyflink`, `%md`.
- No se admite el uso del objeto `z` de [contexto Zeppelin](#). Los métodos que no devuelven nada no harán nada excepto registrar una advertencia. Otros métodos generarán excepciones de Python o no se compilarán en Scala.
- Una nota debe dar como resultado un único trabajo de Apache Flink.
- Las notas con [formularios dinámicos](#) no se admiten para su implementación como una aplicación.
- Los párrafos de `%md` ([Markdown](#)) se omitirán al implementar una aplicación, ya que se espera que contengan documentación legible por humanos que no es adecuada para ejecutarse como parte de la aplicación resultante.
- Los párrafos que estén deshabilitados para ejecutarse en Zeppelin se omitirán al desplegarse como una aplicación. Incluso si un párrafo desactivado utiliza un intérprete incompatible, por ejemplo, `%flink.ipyflink` en una nota con `%flink` and `%flink.ssql` intérpretes, se omitirá al utilizar la nota como una aplicación y no se producirá ningún error.
- Debe haber al menos un párrafo con el código fuente (Flink SQL PyFlink o Flink Scala) que esté habilitado para ejecutarse para que la aplicación se despliegue correctamente.

- La configuración del paralelismo en la directiva del intérprete dentro de un párrafo (por ejemplo `%flink.sql(parallelism=32)`) no se tendrá en cuenta en las aplicaciones que se desplieguen a partir de una nota. En su lugar, puede actualizar la aplicación implementada mediante la AWS API AWS Command Line Interface o la AWS Management Console API para cambiar la configuración del paralelismo o la `ParallelismPer KPU` en función del nivel de paralelismo que requiera la aplicación, o bien puede habilitar el escalado automático de la aplicación implementada.
- Si va a realizar la implementación como una aplicación con estado duradero, su VPC debe tener acceso a Internet. Si su VPC no tiene acceso a Internet, consulte [Implementación como una aplicación con un estado duradero en una VPC sin acceso a Internet](#).

Criterios de Scala/Python

- En tu código de Scala o Python, use el [planificador Blink](#) (`send`, `stenv` para Scala; `s_env`, `st_env` para Python) y no el antiguo planificador "Flink" (`stenv_2` para Scala, `st_env_2` para Python). El proyecto Apache Flink recomienda el uso del planificador de Blink para los casos de uso de producción, que es el planificador predeterminado en Zeppelin y Flink.
- Sus párrafos de Python no deben usar [invocaciones o asignaciones de intérprete de comandos](#) que utilicen `!` o [comandos mágicos de IPython](#), como `%timeit` o `%conda` en notas destinadas a ser implementadas como aplicaciones.
- No puede usar las clases de mayúsculas y minúsculas de Scala como parámetros de funciones que se pasan a operadores de flujo de datos de orden superior, como `map` y `filter`. Para obtener información sobre las clases de casos de Scala, consulte [CLASES DE CASOS](#) en la documentación de Scala.

Criterios SQL

- No se permiten sentencias `SELECT` simples, ya que no hay ningún lugar equivalente a la sección de salida de un párrafo en el que se puedan entregar los datos.
- En cualquier párrafo, las sentencias DDL (`USE`, `CREATE`, `ALTER`, `DROP`, `SET`, `RESET`) deben preceder a las instrucciones DML (`INSERT`). Esto se debe a que las declaraciones DML de un párrafo deben enviarse juntas como un solo trabajo de Flink.
- Debe haber como máximo un párrafo que contenga declaraciones DML. Esto se debe a que, para esta `deploy-as-application` función, solo admitimos el envío de un único trabajo a Flink.

Para obtener más información y un ejemplo, consulte [Traducir, redactar y analizar datos de streaming mediante funciones de SQL con Amazon Managed Service para Apache Flink, Amazon Translate y Amazon Comprehend](#).

Permisos de IAM para blocs de notas de Studio

El servicio gestionado para Apache Flink crea un rol de IAM cuando crea un bloc de notas de Studio mediante AWS Management Console. También asocia a esa función una política que permite el siguiente acceso:

Servicio	Acceso
CloudWatch Registros	Enumeración
Amazon EC2	Enumeración
AWS Glue	Lectura, Escritura
Managed Service para Apache Flink	Leer
Managed Service para Apache Flink V2	Leer
Amazon S3	Lectura, Escritura

Conectores y dependencias

Los conectores permiten leer y escribir datos en diversas tecnologías. El servicio gestionado para Apache Flink incluye tres conectores predeterminados en su bloc de notas de Studio. También puede usar conectores personalizados. Para obtener más información sobre conectores, consulte [Tabla y conectores SQL](#) en la documentación de Apache Flink.

Conectores por defecto

Si utiliza AWS Management Console para crear su bloc de notas de Studio, Managed Service para Apache Flink incluye los siguientes conectores personalizados de forma predeterminada: `flink-sql-connector-flink`, `flink-connector-kafka_2.12` y `aws-msk-iam-auth`. Para crear un bloc de notas de Studio a través de la consola sin estos conectores personalizados,

seleccione la opción Crear con ajustes personalizados. A continuación, cuando llegue a la página de configuraciones, desactive las casillas de verificación situadas junto a los dos conectores.

Si utilizas la [CreateApplication](#) API para crear tu bloc de notas Studio, los `flink-connector-kafka` conectores `flink-sql-connector-flink` y no se incluyen de forma predeterminada. Para añadirlos, debe especificarlos como `MavenReference` en el tipo de datos `CustomArtifactsConfiguration`, como se muestra en los siguientes ejemplos.

El conector `aws-msk-iam-auth` es el conector que se utilizará con Amazon MSK e incluye la característica de autenticarse automáticamente con IAM.

Note

Las versiones de conector que se muestran en el siguiente ejemplo son las únicas versiones que admitimos.

For the Kinesis connector:

```
"CustomArtifactsConfiguration": [{
  "ArtifactType": "DEPENDENCY_JAR",
  "MavenReference": {
    "GroupId": "org.apache.flink",

    "ArtifactId": "flink-sql-connector-kinesis",
    "Version": "1.15.4"

  }
}]
```

For authenticating with AWS MSK through AWS IAM:

```
"CustomArtifactsConfiguration": [{
  "ArtifactType": "DEPENDENCY_JAR",
  "MavenReference": {
    "GroupId": "software.amazon.msk",
    "ArtifactId": "aws-msk-iam-auth",
    "Version": "1.1.6"

  }
}]
```

For the Apache Kafka connector:

```
"CustomArtifactsConfiguration": [{
  "ArtifactType": "DEPENDENCY_JAR",
  "MavenReference": {
    "GroupId": "org.apache.flink",

    "ArtifactId": "flink-connector-kafka",
    "Version": "1.15.4"

  }
}]
```

Para añadir estos conectores a un bloc de notas existente, usa la operación [UpdateApplicationAPI](#) y especifícalos como `MavenReference` en el tipo de `CustomArtifactsConfigurationUpdate` datos.

Note

Puede establecer el valor `true failOnError` para el conector `flink-sql-connector-kinesis` en la API de la tabla.

Dependencias y conectores personalizados

Para usar AWS Management Console para añadir una dependencia o un conector personalizado a tu portátil Studio, siga estos pasos:

1. Cargue el archivo del conector personalizado en Amazon S3.
2. En AWS Management Console, elija la opción creación personalizada para crear su bloc de notas de Studio.
3. Siga el flujo de trabajo de creación de blocs de notas de Studio hasta llegar al paso de configuración.
4. En la sección Conectores personalizados, seleccione Añadir conector personalizado.
5. Especifique la ubicación de Amazon S3 de la dependencia o del conector personalizado.
6. Elija Guardar cambios.

Para añadir un JAR de dependencia o un conector personalizado al crear un nuevo bloc de notas de Studio mediante la [CreateApplicationAPI](#), especifique la ubicación en Amazon S3 del JAR de

dependencia o del conector personalizado en el tipo de `CustomArtifactsConfiguration` datos. Para añadir una dependencia o un conector personalizado a un bloc de notas de Studio existente, invoque la operación de [UpdateApplication](#) API y especifique la ubicación en Amazon S3 del JAR de la dependencia o del conector personalizado en el tipo de `CustomArtifactsConfigurationUpdate` datos.

Note

Cuando incluye una dependencia o un conector personalizado, también debe incluir todas sus dependencias transitivas que no estén incluidas en él.

Funciones definidas por el usuario

Las funciones definidas por el usuario (User-defined functions, UDF) son puntos de extensión que permiten llamar a la lógica de uso frecuente o a la lógica personalizada que no se puede expresar de otro modo en las consultas. Puede usar Python o un lenguaje JVM como Java o Scala para implementar sus UDF en párrafos dentro de su bloc de notas de Studio. También puede añadir a tu bloc de notas de Studio archivos JAR externos que contengan UDF implementadas en un lenguaje JVM.

Al implementar archivos JAR que registren clases abstractas que subclasifican `UserDefinedFunction` (o sus propias clases abstractas), use el alcance proporcionado en Apache Maven, las declaraciones de dependencia `compileOnly` en Gradle, el alcance proporcionado en SBT o una directiva equivalente en la configuración de compilación del proyecto UDF. Esto permite que el código fuente de la UDF se compile con las API de Flink, pero las propias clases de la API de Flink no se incluyen en los artefactos de compilación. Consulte este [pom](#) del ejemplo jar UDF, que cumple con este requisito previo en un proyecto de Maven.

Note

Para ver un ejemplo de configuración, consulte [Traducir, redactar y analizar datos de streaming mediante funciones SQL con Amazon Managed Service para Apache Flink, Amazon Translate y Amazon Comprehend](#) en el AWSblog de Machine Learning.

Para usar la consola para añadir archivos JAR de la UDF a su bloc de notas de Studio, siga estos pasos:

1. Cargue su archivo JAR UDF en Amazon S3.
2. En AWS Management Console, elija la opción creación personalizada para crear su bloc de notas de Studio.
3. Siga el flujo de trabajo de creación de blocs de notas de Studio hasta llegar al paso de Configuración.
4. En la sección Funciones definidas por el usuario, seleccione Añadir función definida por el usuario.
5. Especifique la ubicación en Amazon S3 del archivo JAR o el archivo ZIP que contiene la implementación de su UDF.
6. Elija Guardar cambios.

Para añadir un JAR de UDF al crear un nuevo bloc de notas de Studio mediante la [CreateApplicationAPI](#), especifique la ubicación del JAR en el tipo de `CustomArtifactConfiguration` datos. Para añadir un JAR de UDF a un bloc de notas de Studio existente, invoca la operación de la [UpdateApplicationAPI](#) y especifica la ubicación del JAR en el `CustomArtifactsConfigurationUpdate` tipo de datos. Alternativamente, puede usar AWS Management Console para agregar archivos JAR UDF a su bloc de notas de Studio.

Consideraciones con funciones definidas por el usuario

- El servicio gestionado para Apache Flink Studio utiliza la [terminología de Apache Zeppelin](#), según la cual un bloc de notas es una instancia de Zeppelin que puede contener varias notas. De este modo, cada nota puede contener varios párrafos. Con Managed Service para Apache Flink Studio, el proceso de interpretación se comparte entre todas las notas del bloc de notas. Por lo tanto, si realizas un registro de funciones explícito con [createTemporarySystemFunction](#) en una nota, podrás hacer referencia a la misma tal cual en otra nota del mismo cuaderno.

Sin embargo, la operación Implementar como aplicación funciona en una nota individual y no en todas las notas del bloc de notas. Al realizar la implementación como aplicación, solo se utiliza el contenido de la nota activa para generar la aplicación. Cualquier registro de funciones explícito realizado en otros blocs de notas no forma parte de las dependencias de aplicación generadas. Además, durante la opción Implementar como aplicación, se produce un registro implícito de la función al convertir el nombre de la clase principal de JAR en una cadena en minúsculas.

Por ejemplo, si `TextAnalyticsUDF` es la clase principal de JAR de UDF, un registro implícito dará como resultado el nombre de la función `textanalyticsudf`. Por lo tanto, si el registro

explícito de una función en la nota 1 de Studio se produce de la siguiente manera, todas las demás notas de ese bloc de notas (por ejemplo, la nota 2) pueden hacer referencia a la función por su nombre `myNewFuncNameForClass` gracias al intérprete compartido:

```
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new
TextAnalyticsUDF())
```

Sin embargo, durante la implementación como operación de aplicación (nota 2), este registro explícito no se incluirá en las dependencias y, por lo tanto, la aplicación implementada no funcionará según lo esperado. Debido al registro implícito, de forma predeterminada se espera que todas las referencias a esta función aparezcan con `textanalyticsudf` y no `myNewFuncNameForClass`.

Si es necesario registrar el nombre de una función personalizada, se espera que la propia nota 2 contenga otro párrafo para realizar otro registro explícito de la siguiente manera:

```
%flink(parallelism=1)
import com.amazonaws.kinesis.udf.textanalytics.TextAnalyticsUDF
# re-register the JAR for UDF with custom name
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new TextAnalyticsUDF())
```

```
%flink. sql(type=update, parallelism=1)
INSERT INTO
    table2
SELECT
    myNewFuncNameForClass(column_name)
FROM
    table1
;
```

- Si su archivo JAR de UDF incluye los SDK de Flink, configure su proyecto de Java para que el código fuente de la UDF se pueda compilar con los SDK de Flink, pero las clases del SDK de Flink no estén incluidas en el artefacto de compilación, por ejemplo, el JAR.

Puede usar el alcance `provided` en Apache Maven, las declaraciones de dependencia `compileOnly` en Gradle, el alcance `provided` en SBT o una directiva equivalente en la configuración de compilación de sus proyectos UDF. Puede consultar este [pom](#) del ejemplo jar UDF, que cumple con este requisito previo en un proyecto de Maven. Para ver un step-by-step tutorial completo, consulte este tutorial sobre cómo [traducir, redactar y analizar datos de streaming](#)

[mediante funciones de SQL con Amazon Managed Service para Apache Flink, Amazon Translate y Amazon Comprehend.](#)

Activación de puntos de control

Los puntos de control se activan mediante la configuración del entorno. Para obtener información sobre los puntos de control, consulte la guía para desarrolladores de [Fault Tolerance](#) en [Managed Service para Apache Flink](#).

Establecer el intervalo de puntos de control

El siguiente ejemplo de código de Scala establece el intervalo de puntos de control de la aplicación en un minuto:

```
// start a checkpoint every 1 minute
stenv.enableCheckpointing(60000)
```

El siguiente ejemplo de código de Python establece el intervalo de puntos de control de la aplicación en un minuto:

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.interval", "1min"
)
```

Establecer el tipo de punto de control

El siguiente ejemplo de código de Scala establece el modo de punto de control de la aplicación en EXACTLY_ONCE (predeterminado):

```
// set mode to exactly-once (this is the default)
stenv.getCheckpointConfig.setCheckpointingMode(CheckpointingMode.EXACTLY_ONCE)
```

El siguiente ejemplo de código de Python establece el modo de punto de control de la aplicación en EXACTLY_ONCE (predeterminado):

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.mode", "EXACTLY_ONCE"
)
```

Trabajar con AWS Glue

El bloc de notas de Studio almacena y obtiene información sobre sus orígenes de datos y receptores de AWS Glue. Al crear el bloc de notas de Studio, se especifica la base de datos AWS Glue que contiene la información de conexión. Al acceder a las fuentes y receptores de datos, se especifican las tablas AWS Glue contenidas en la base de datos. Las tablas AWS Glue proporcionan acceso a las conexiones AWS Glue que definen las ubicaciones, los esquemas y los parámetros de las fuentes de datos y los destinos.

Los blocs de notas de Studio utilizan las propiedades de las tablas para almacenar datos específicos de la aplicación. Para obtener más información, consulte [Propiedades de la tabla](#).

Para ver un ejemplo de cómo configurar una conexión AWS Glue, una base de datos y una tabla para utilizarlas con las libretas de Studio, consulte [Creación de una base de datos de AWS Glue](#) en el tutorial de [Tutorial sobre cómo crear un bloc de notas de Studio](#).

Propiedades de la tabla

Además de los campos de datos, las tablas AWS Glue proporcionan otra información al bloc de notas de Studio mediante las propiedades de las tablas. Managed Service para Apache Flink utiliza las siguientes propiedades de tabla AWS Glue:

- [Uso de valores de tiempo de Apache Flink](#): estas propiedades definen cómo Managed Service para Apache Flink emite los valores de tiempo de procesamiento de datos internos de Apache Flink.
- [Uso de Flink Connector y propiedades de formato](#): estas propiedades proporcionan información sobre sus flujos de datos.

Para agregar una propiedad a una tabla AWS Glue, realice lo siguiente:

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en <https://console.aws.amazon.com/glue/>.
2. En la lista de tablas, elija aquella que la aplicación utiliza para almacenar la información de conexión de datos. Seleccione Acción y edite los detalles de la tabla.
3. En Propiedades de la tabla, introduzca **managed-flink.proctime** para la clave y **user_action_time** para el valor.

Uso de valores de tiempo de Apache Flink

Apache Flink proporciona valores de tiempo que describen cuándo se produjeron los eventos de procesamiento de la transmisión, como el [Tiempo de procesamiento](#) y el [Tiempo del evento](#). Para incluir estos valores en el resultado de la aplicación, defina las propiedades en la tabla AWS Glue que indiquen al tiempo de ejecución de Managed Service para Apache Flink que emita estos valores en los campos especificados.

Las claves y los valores que utiliza en las propiedades de la tabla son los siguientes:

Tipo de timestamp	Clave	Valor
Tiempo de procesamiento	managed-flink.proctime	The column name that AWS Glue will use to expose the value. This column name does not correspond to an existing table column.
Tiempo del evento	managed-flink.rowtime	The column name that AWS Glue will use to expose the value. This column name corresponds to an existing table column.
	managed-flink.watermark. <i>column_name</i> .milliseconds	The watermark interval in milliseconds

Uso de Flink Connector y propiedades de formato

La información sobre las fuentes de datos se proporciona a los conectores Flink de la aplicación mediante las propiedades de tabla AWS Glue. A continuación se muestran algunos ejemplos de las propiedades que Managed Service para Apache Flink utiliza para los conectores:

Tipo de conector	Clave	Valor
Kafka	<code>format</code>	The format used to deserialize and serialize Kafka messages, e.g. <code>json</code> or <code>csv</code> .
	<code>scan.startup.mode</code>	The startup mode for the Kafka consumer, e.g. <code>earliest-offset</code> or <code>Marca de tiempo</code> .
Kinesis	<code>format</code>	The format used to deserialize and serialize Kinesis data stream records, e.g. <code>json</code> or <code>csv</code> .
	<code>aws.region</code>	The AWS region where the stream is defined.
S3 (Filesystem)	<code>format</code>	The format used to deserialize and serialize files, e.g. <code>json</code> or <code>csv</code> .
	<code>ruta</code>	The Amazon S3 path, e.g. <code>s3://mybucket/</code> .

Para obtener más información sobre otros conectores además de Kinesis y Apache Kafka, consulte la documentación del conector.

Ejemplos y tutoriales

Temas

- [Tutorial: Creación de un bloc de notas de Studio en Managed Service para Apache Flink](#)
- [Tutorial: Implementación como una aplicación con estado duradero](#)
- [Ejemplos](#)

Tutorial: Creación de un bloc de notas de Studio en Managed Service para Apache Flink

El siguiente tutorial muestra cómo crear un bloc de notas de Studio que lea datos de un clúster de Flujo de Datos de Kinesis o Amazon MSK.

Este tutorial contiene las siguientes secciones:

- [Configuración](#)
- [Creación de una base de datos de AWS Glue](#)
- [Pasos siguientes](#)
- [Creación de un bloc de notas de Studio con Kinesis Data Streams](#)
- [Creación de un bloc de notas de Studio con Amazon MSK](#)
- [Limpiar su aplicación y los recursos dependientes](#)

Configuración

Asegúrese de que su AWS CLI sea versión 2 o una posterior. Para instalar el AWS CLI más reciente, consulte [Instalación, actualización y desinstalación de AWS CLI versión 2](#).

Creación de una base de datos de AWS Glue

Su bloc de notas de Studio utiliza una base de datos [AWS Glue](#) para los metadatos sobre su origen de datos de Amazon MSK.

Creación de una base de datos de AWS Glue

1. Abra la consola de AWS Glue en <https://console.aws.amazon.com/glue/>.
2. Seleccione Agregar una base de datos. En la ventana Añadir base de datos, introduzca **default** en el nombre de la base de datos. Seleccione Crear.

Pasos siguientes

Con este tutorial, puede crear un bloc de notas de Studio que utilice flujos de datos de Kinesis o Amazon MSK:

- [Kinesis Data Streams](#) : con flujos de datos de Kinesis, puede crear rápidamente una aplicación que utilice un flujo de datos de Kinesis como origen. Solo necesita crear un flujo de datos de Kinesis como recurso dependiente.
- [Amazon MSK](#) : con Amazon MSK, cree una aplicación que utiliza un clúster de Amazon MSK como origen. Debe crear una Amazon VPC, una instancia de cliente de Amazon EC2 y un clúster de Amazon MSK como recursos dependientes.

Creación de un bloc de notas de Studio con Kinesis Data Streams

En este tutorial se describe cómo crear un bloc de notas de Studio que utilice un flujo de datos de Kinesis como fuente.

Este tutorial contiene las siguientes secciones:

- [Configuración](#)
- [Creación de una tabla de AWS Glue](#)
- [Cree un bloc de notas de Studio con Kinesis Data Streams](#)
- [Envíe datos a su flujo de datos de Kinesis](#)
- [Pruebe su bloc de notas de Studio](#)

Configuración

Antes de crear un bloc de notas de Studio, cree un flujo de datos de Kinesis (ExampleInputStream). Su aplicación utiliza estos flujos para el origen de la aplicación.

Puede crear este flujo mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne el nombre **ExampleInputStream** al flujo y establezca el número de particiones abiertas en **1**.

Para crear el flujo (ExampleInputStream), utilizando el AWS CLI, utilice el siguiente comando de Amazon Kinesis create-stream AWS CLI.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-east-1 \  

```

```
--profile adminuser
```

Creación de una tabla de AWS Glue

Su bloc de notas Studio utiliza una [AWS Glue](#) base de datos para los metadatos sobre su origen de datos de Kinesis Data Streams.

Note

Puede crear primero la base de datos manualmente o dejar que Managed Service para Apache Flink la cree por usted al crear el bloc de notas. De forma similar, puede crear la tabla manualmente tal y como se describe en esta sección o puede usar el código conector de creación de tablas para Managed Service para Apache Flink en su bloc de notas en Apache Zeppelin para crear la tabla mediante una instrucción DDL. A continuación, puede comprobar AWS Glue para asegurarse de que la tabla se creó correctamente.

Crear una tabla

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en <https://console.aws.amazon.com/glue/>.
2. Si aún no tiene una AWS Glue base de datos, elija Bases de datos en la barra de navegación izquierda. Elija Agregar una base de datos. En la ventana Añadir base de datos, introduzca **default** en el nombre de la base de datos. Seleccione Crear.
3. En la barra de navegación izquierda, seleccione Tablas. En la página Tablas, seleccione Añadir tablas y Añadir tabla manualmente.
4. En la página Configurar las propiedades de la tabla, introduzca **stock** como Nombre de la tabla. Asegúrese de seleccionar la base de datos que creó anteriormente. Elija Siguiente.
5. En la página Añadir un almacén de datos, elija Kinesis. Para el Nombre del flujo, introduzca **ExampleInputStream**. Para la URL de origen de Kinesis, pulse Intro **https://kinesis.us-east-1.amazonaws.com**. Si copia y pega la URL de origen de Kinesis, asegúrese de eliminar los espacios iniciales o finales. Elija Siguiente.
6. En la página de Clasificación, seleccione JSON. Elija Siguiente.
7. En la página Definir un esquema, elija Añadir columna para añadir una. Añada columnas con las siguientes propiedades:

Nombre de la columna	Tipo de datos
ticker	cadena
precio	double

Elija Siguiente.

8. En la página siguiente, verifique su configuración y seleccione Finalizar.
9. Elija la tabla recién creada de la lista de tablas.
10. Elija Editar tabla y añada una propiedad con la clave `managed-flink.proctime` y el valor `proctime`.
11. Seleccione Aplicar.

Cree un bloc de notas de Studio con Kinesis Data Streams


Ahora que ha creado los recursos que utiliza su aplicación, crea su bloc de notas de Studio.

Para crear la aplicación, puede utilizar el AWS Management Console o el AWS CLI.

- [Cree un bloc de notas de Studio con el AWS Management Console](#)
- [Cree un bloc de notas de Studio con el AWS CLI](#)

Cree un bloc de notas de Studio con el AWS Management Console

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard>.
2. En la página de las aplicaciones Managed Service para Apache Flink, seleccione la pestaña Studio. Seleccione Crear bloc de notas de Studio.

 Note

También puede crear un bloc de notas de Studio desde las consolas de Amazon MSK o Kinesis Data Streams seleccionando el clúster de Amazon MSK o la el flujo de datos de Kinesis de entrada y eligiendo Procesar datos en tiempo real.

3. En la página Crear bloc de notas de Studio, proporcione la siguiente información:

- Introduzca **MyNotebook** como nombre del bloc de notas.
- Elija el valor predeterminado para la AWSbase de datos de Glue.

Seleccione Crear bloc de notas de Studio.

4. En la MyNotebookpágina, selecciona Ejecutar. Espere a que el Estado muestre En ejecución. Se aplican cargos cuando el bloc de notas se está ejecutando.

Cree un bloc de notas de Studio con el AWS CLI

Para crear su bloc de notas de Studio usando elAWS CLI, haga lo siguiente:

1. Verifique su ID de cuenta. Necesitará este valor para crear su aplicación.
2. Cree el rol `arn:aws:iam::AccountID:role/ZeppeleinRole` y añada los siguientes permisos al rol creado automáticamente por consola.

```
"kinesis:GetShardIterator",
```

```
"kinesis:GetRecords",
```

```
"kinesis:ListShards"
```

3. Cree un archivo denominado `create.json` con el siguiente contenido. Reemplace los valores de marcador de posición con su información.

```
{
  "ApplicationName": "MyNotebook",
  "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
  "ApplicationMode": "INTERACTIVE",
  "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppeleinRole",
  "ApplicationConfiguration": {
    "ApplicationSnapshotConfiguration": {
      "SnapshotsEnabled": false
    },
    "ZeppelinApplicationConfiguration": {
      "CatalogConfiguration": {
        "GlueDataCatalogConfiguration": {
          "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/
default"
        }
      }
    }
  }
}
```

```
    }  
  }  
}
```

4. Para crear su aplicación, ejecute el siguiente comando:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create.json
```

5. Una vez completado el comando, verá un resultado que muestra los detalles de su nuevo bloc de notas de Studio. A continuación se muestra un ejemplo de la salida.

```
{  
  "ApplicationDetail": {  
    "ApplicationARN": "arn:aws:kinesisanalyticstv2-east-1:012345678901:application/MyNotebook",  
    "ApplicationName": "MyNotebook",  
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",  
    "ApplicationMode": "INTERACTIVE",  
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",  
    ...  
  }  
}
```

6. Para iniciar su aplicación, ejecute el siguiente comando. Sustituya los valores de muestra por su ID de la cuenta.

```
aws kinesisanalyticstv2 start-application --application-arn  
arn:aws:kinesisanalyticstv2-east-1:012345678901:application/MyNotebook\
```

Envíe datos a su flujo de datos de Kinesis

Para enviar los datos de prueba al flujo de datos de Kinesis, haga lo siguiente:

1. Abra el [Kinesis Data Generator](#).
2. Elija Crear un usuario de Cognito con. CloudFormation
3. La consola AWS CloudFormation se abre con la plantilla de Kinesis Data Generator. Elija Siguiente.
4. En la página Especificar detalles de pila, ingrese el nombre y la contraseña de su usuario de Cognito. Elija Siguiente.
5. En la página Configurar opciones de pila, elija Siguiente.

6. En la página Revisar Kinesis-Data-Generator-Cognito-User, elija Acepto que podría crear recursos de IAM. AWS CloudFormation casilla de verificación. Elija Crear pila.
7. Espere a que termine la creación de la pila de AWS CloudFormation. Una vez completada la pila, abra la pila Kinesis-Data-Generator-Cognito-User en la consola AWS CloudFormation y seleccione la pestaña Salidas. Abra la URL que aparece para el valor KinesisDataGeneratorUrlde salida.
8. En la página de Amazon Kinesis Data Generator, inicie sesión con las credenciales que creó en el paso 4.
9. En la siguiente página , especifique los valores siguientes:

Región	us-east-1
Flujo/Flujo de Kinesis Data Firehose	ExampleInputStream
Registros por segundo	1

En Plantilla de registro, pegue el siguiente código:

```
{
  "ticker": "{{random.arrayElement(
    ["AMZN", "MSFT", "GOOG"]
  )}}",
  "price": {{random.number(
    {
      "min":10,
      "max":150
    }
  )}}
}
```

10. Elija Enviar datos.
11. El generador enviará los datos a su flujo de datos de Kinesis.

Deje el generador ejecutándose mientras completa la siguiente sección.

Pruebe su bloc de notas de Studio

En esta sección, utilizará su bloc de notas de Studio para consultar datos de su flujo de datos de Kinesis.

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard>.
2. En la página de las aplicaciones Managed Service para Apache Flink, seleccione la pestaña Bloc de notas de Studio. Elige MyNotebook.
3. En la MyNotebook página, elija Abrir en Apache Zeppelin.

La interfaz de Apache Zeppelin se abre en una pestaña nueva.

4. En la página ¡Bienvenido a Zeppelin!, elija la Zeppelin Note.
5. En la página Zeppelin Note, introduzca la siguiente consulta en una nota nueva:

```
%flink.ssql(type=update)
select * from stock
```

Seleccione el icono de reproducción.

Al cabo de poco tiempo, el bloc de notas muestra los datos de la flujo de datos de Kinesis.

Para abrir el Panel de control de Apache Flink de su aplicación y ver los aspectos operativos, elija TRABAJO DE FLINK. Para obtener más información sobre el Panel de control de Flink, consulte el [Panel de control de Apache Flink](#) en la Guía para desarrolladores de [Managed Service para Apache Flink](#).

Para ver más ejemplos de consultas SQL de Flink Streaming, consulte [Consultas](#) en la [documentación de Apache Flink](#).

Creación de un bloc de notas de Studio con Amazon MSK

En este tutorial se describe cómo crear un bloc de notas de Studio que utilice un clúster de Amazon MSK como fuente.

Este tutorial contiene las siguientes secciones:

- [Configuración](#)
- [Agregue una puerta de enlace NAT a su VPC](#)

- [Cree una conexión y tabla de AWS Glue](#)
- [Cree un bloc de notas de Studio con Amazon MSK](#)
- [Envíe datos a su clúster de Amazon MSK](#)
- [Pruebe su bloc de notas de Studio](#)

Configuración

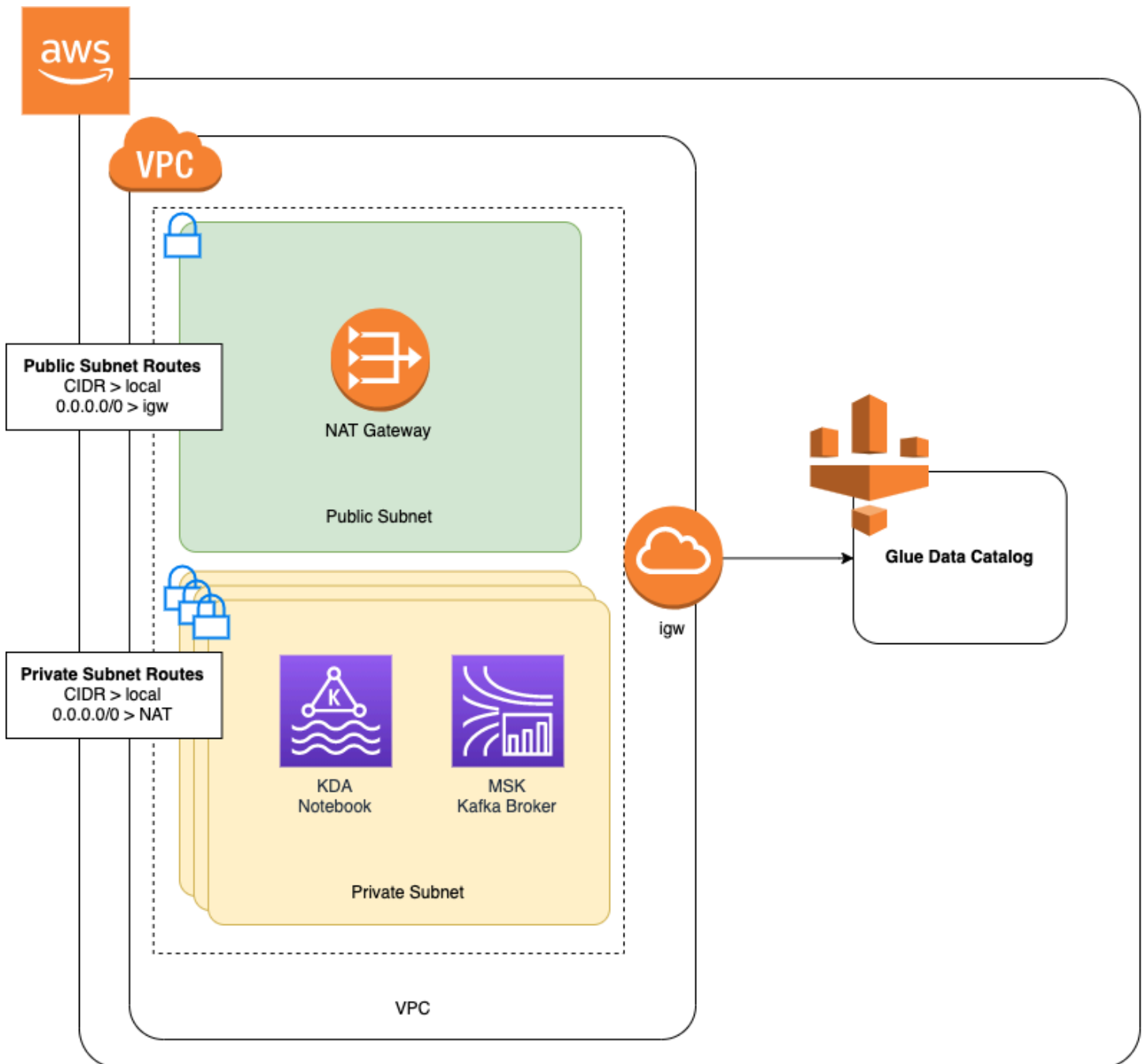
Para este tutorial, necesita un clúster de Amazon MSK que permita el acceso a texto sin formato. Si aún no ha configurado un clúster de Amazon MSK, siga el tutorial [Cómo empezar a utilizar Amazon MSK](#) para crear una Amazon VPC, un clúster de Amazon MSK, un tema y una instancia de cliente de Amazon EC2.

Al seguir el tutorial, haga lo siguiente:

- En el [paso 3: cree un clúster de Amazon MSK](#), en el paso 4, cambie el `ClientBroker` valor de TLS a **PLAINTEXT**.

Agregue una puerta de enlace NAT a su VPC

Si ha creado un clúster de Amazon MSK siguiendo el tutorial [Cómo empezar a utilizar Amazon MSK](#), o si su Amazon VPC existente aún no tiene una puerta de enlace NAT para sus subredes privadas, debe añadir una puerta de enlace NAT a su Amazon VPC. En el siguiente diagrama se muestra la arquitectura.



Para crear una puerta de enlace NAT para su Amazon VPC, haga lo siguiente:

1. Abra la consola de Amazon VPC en <https://console.aws.amazon.com/vpc/>.
2. En la barra de navegación izquierda, elija puertas de enlace NAT.
3. En la página puertas de enlace NAT, seleccione Crear puerta de enlace NAT.
4. En la página Crear puerta de enlace NAT, especifique los valores siguientes:

Nombre: optional	ZeppelinGateway
Subred	AWSKafkaTutorialSubnet1
Identificación de asignación de dirección IP elástica	Choose an available Elastic IP. If there are no Elastic IPs available, choose Asignación de dirección IP elástica, and then choose the Elastic IP that the console creates.

Elija Create a NAT Gateway (Crear una puerta de enlace NAT).

5. En la de navegación izquierda, elija Tablas de ruteo.
6. Elija Create Route Table (Crear tabla de ruteo).
7. En la página Crear tabla de enrutamiento, proporcione la siguiente información:
 - Name tag: **ZeppelinRouteTable**
 - VPC: elija su VPC (por ejemplo, AWSKafkaTutorialVPC).

Seleccione Create (Crear).

8. En la lista de tablas de enrutamiento, elija ZeppelinRouteTable. Elija la pestaña Rutas y, a continuación, Editar rutas.
9. En la pestaña Editar rutas, elija Añadir rutas.
10. En Para Destino, escriba **0.0.0.0/0**. Para Destino, elija puerta de enlace NAT, ZeppelinGateway. Elija Guardar rutas. Elija Cerrar.
11. En la página de Tabla de enrutamiento, con ZeppelinRouteTable seleccionada, elija la pestaña Asociaciones de subredes. Elija Editar asociaciones de subredes.
12. En la página Editar asociaciones de subredes, elija AWSKafkaTutorialSubnet2 y AWSKafkaTutorialSubnet3. Seleccione Guardar.

Cree una conexión y tabla de AWS Glue

Su bloc de notas de Studio utiliza una base de datos [AWS Glue](#) para los metadatos sobre su origen de datos de Amazon MSK. En esta sección, crea una AWS Glue conexión que describe cómo

acceder a su clúster de Amazon MSK y una tabla AWS Glue que describe cómo presentar los datos de su origen de datos a clientes como su bloc de notas Studio.

Crear una conexión

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en <https://console.aws.amazon.com/glue/>.
2. Si aún no tiene una AWS Glue base de datos, elija Bases de datos en la barra de navegación izquierda. Elija Agregar una base de datos. En la ventana Añadir base de datos, introduzca **default** en el nombre de la base de datos. Seleccione Crear.
3. En la barra de navegación de la izquierda, seleccione Conexiones. Elija Añadir conexión.
4. En la ventana Añadir conexión, introduzca los siguientes valores:
 - En Nombre de conexión, ingrese **ZeppelinConnection**.
 - En Tipo de conexión, elija Kafka.
 - En las URL del servidor de arranque de Kafka, proporcione la cadena del agente de arranque del clúster. Puede obtener los agentes de arranque desde la consola MSK o ingresando el siguiente comando de la CLI:

```
aws kafka get-bootstrap-brokers --region us-east-1 --cluster-arn ClusterArn
```

- Desactive la casilla de verificación Exigir conexión SSL.

Elija Siguiente.

5. En la página VPC, especifique los valores siguientes:
 - Para VPC, elija el nombre de su VPC (por ejemplo, AWSKafkaTutorialVPC).
 - En Subred, elija AWSKafkaTutorialSubnet2.
 - Para los grupos de seguridad, elija todos los grupos disponibles.

Elija Siguiente.

6. En la página Propiedades de la conexión o Acceso a la conexión, seleccione Finalizar.

Crear una tabla

Note

Puede crear la tabla manualmente tal y como se describe en los pasos siguientes, o puede usar el código conector de creación de tablas para Managed Service para Apache Flink en su bloc de notas en Apache Zeppelin para crear la tabla mediante una instrucción DDL. A continuación, puede comprobar AWS Glue para asegurarse de que la tabla se creó correctamente.

1. En la barra de navegación izquierda, seleccione Tablas. En la página Tablas, seleccione Añadir tablas y Añadir tabla manualmente.
2. En la página Configurar las propiedades de la tabla, introduzca **stock** como Nombre de la tabla. Asegúrese de seleccionar la base de datos que creó anteriormente. Elija Siguiente.
3. En la página Añadir almacén de datos, elija Kafka. Para el Nombre del tema, introduzca el nombre del tema (por ejemplo, AWSKafkaTutorialTopic). Para Conexión, elija ZeppelinConnection.
4. En la página de Clasificación, seleccione JSON. Elija Siguiente.
5. En la página Definir un esquema, elija Añadir columna para añadir una. Añada columnas con las siguientes propiedades:

Nombre de la columna	Tipo de datos
ticker	string
price	double

Elija Siguiente.

6. En la página siguiente, verifique su configuración y seleccione Finalizar.
7. Elija la tabla recién creada de la lista de tablas.
8. Elija Editar tabla y añada una propiedad con la clave `managed-flink.proctime` y el valor `proctime`.
9. Seleccione Aplicar.

Cree un bloc de notas de Studio con Amazon MSK

Ahora que ha creado los recursos que utiliza su aplicación, crea su bloc de notas de Studio.

Puede crear su aplicación mediante AWS Management Console o AWS CLI.

- [Cree un bloc de notas de Studio con el AWS Management Console](#)
- [Cree un bloc de notas de Studio con el AWS CLI](#)

Note

También puede crear un bloc de notas de Studio desde la consola Amazon MSK seleccionando un clúster existente y, a continuación, seleccionando Procesar datos en tiempo real.

Cree un bloc de notas de Studio con el AWS Management Console

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard>.
2. En la página de aplicaciones de Managed Service para Apache Flink, seleccione la pestaña Studio. Seleccione Crear bloc de notas de Studio.

Note

Para crear un bloc de notas de Studio desde las consolas Amazon MSK o Kinesis Data Streams, seleccione el clúster Amazon MSK o el flujo de datos de Kinesis de entrada y, a continuación, elija Procesar datos en tiempo real.

3. En la página Crear bloc de notas de Studio, proporcione la siguiente información:
 - Introduzca **MyNotebook** como Nombre del bloc de notas de Studio.
 - Elija el valor predeterminado para la Base de datos de Glue de AWS.

Seleccione Crear bloc de notas de Studio.

4. En la página MyNotebook, seleccione la pestaña Configuración. En la sección Redes, elija Editar.

5. En la página Editar redes para MyNotebook, elija la configuración de VPC basada en el clúster de Amazon MSK. Elija su clúster de Amazon MSK para el Clúster de Amazon MSK. Elija Guardar cambios.
6. En la página MyNotebook, seleccione Ejecutar. Espere a que el Estado muestre En ejecución.

Cree un bloc de notas de Studio con el AWS CLI

Para crear su bloc de notas de Studio usando elAWS CLI, haga lo siguiente:

1. Verifique que disponga de la siguiente información. Necesita estos valores para crear su aplicación.
 - Su ID de cuenta de .
 - Los ID de la subred y el ID de grupo de seguridad de Amazon VPC que contiene su clúster de Amazon MSK.
2. Cree un archivo denominado `create.json` con el siguiente contenido. Reemplace los valores de marcador de posición con su información.

```
{
  "ApplicationName": "MyNotebook",
  "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
  "ApplicationMode": "INTERACTIVE",
  "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
  "ApplicationConfiguration": {
    "ApplicationSnapshotConfiguration": {
      "SnapshotsEnabled": false
    },
    "VpcConfigurations": [
      {
        "SubnetIds": [
          "SubnetID 1",
          "SubnetID 2",
          "SubnetID 3"
        ],
        "SecurityGroupIds": [
          "VPC Security Group ID"
        ]
      }
    ],
    "ZeppelinApplicationConfiguration": {
```

```

    "CatalogConfiguration": {
      "GlueDataCatalogConfiguration": {
        "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/
default"
      }
    }
  }
}

```

3. Para crear su aplicación, ejecute el siguiente comando:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create.json
```

4. Una vez completado el comando, debería ver un resultado similar al siguiente, con los detalles de su nuevo bloc de notas de Studio:

```

{
  "ApplicationDetail": {
    "ApplicationARN": "arn:aws:kinesisanalyticstv2:us-east-1:012345678901:application/MyNotebook",
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
    ...
  }
}

```

5. Para iniciar su aplicación, ejecute el siguiente comando. Sustituya los valores de muestra por su ID de la cuenta.

```
aws kinesisanalyticstv2 start-application --application-arn
arn:aws:kinesisanalyticstv2:us-east-1:012345678901:application/MyNotebook\
```

Envíe datos a su clúster de Amazon MSK

En esta sección, ejecuta un script de Python en su cliente Amazon EC2 para enviar datos a su origen de datos de Amazon MSK.

1. Conecte con su cliente Amazon EC2.
2. Ejecute los siguientes comandos para instalar la versión 3 de Python, Pip y el paquete Kafka para Python, y confirme las acciones:

```
sudo yum install python37
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
pip install kafka-python
```

3. Configure el AWS CLI en su equipo cliente con el siguiente comando:

```
aws configure
```

Proporcione las credenciales de su cuenta y **us-east-1** para region.

4. Cree un archivo denominado `stock.py` con el siguiente contenido. Sustituya el valor del ejemplo por la cadena de agentes de arranque de su clúster de Amazon MSK y actualice el nombre del tema si este no es `AWSKafkaTutorialTopic`:

```
from kafka import KafkaProducer
import json
import random
from datetime import datetime

BROKERS = "<<Bootstrap Broker List>>"
producer = KafkaProducer(
    bootstrap_servers=BROKERS,
    value_serializer=lambda v: json.dumps(v).encode('utf-8'),
    retry_backoff_ms=500,
    request_timeout_ms=20000,
    security_protocol='PLAINTEXT')

def getStock():
    data = {}
    now = datetime.now()
    str_now = now.strftime("%Y-%m-%d %H:%M:%S")
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data

while True:
    data =getStock()
```

```
# print(data)
try:
    future = producer.send("AWSKafkaTutorialTopic", value=data)
    producer.flush()
    record_metadata = future.get(timeout=10)
    print("sent event to Kafka! topic {} partition {} offset
{}".format(record_metadata.topic, record_metadata.partition,
record_metadata.offset))
except Exception as e:
    print(e.with_traceback())
```

5. Ejecute el script con el siguiente comando:

```
$ python3 stock.py
```

6. Deje el script en ejecución mientras completa la siguiente sección.

Pruebe su bloc de notas de Studio

En esta sección, utilizará su bloc de notas de Studio para consultar datos de su clúster de Amazon MSK.

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard>.
2. En la página de las aplicaciones de Managed Service para Apache Flink, seleccione la pestaña Bloc de notas de Studio. Elija MyNotebook.
3. En la página MyNotebook, seleccione Abrir en Apache Zeppelin.

La interfaz de Apache Zeppelin se abre en una pestaña nueva.

4. En la página ¡Bienvenido a Zeppelin!, elija la nueva nota de Zeppelin.
5. En la página Zeppelin Note, introduzca la siguiente consulta en una nota nueva:

```
%flink.ssql(type=update)
select * from stock
```

Seleccione el icono de reproducción.

La aplicación muestra los datos del clúster de Amazon MSK.

Para abrir el Panel de control de Apache Flink de su aplicación y ver los aspectos operativos, elija TRABAJO DE FLINK. Para obtener más información sobre el Panel de control de Flink, consulte el [Panel de control de Apache Flink](#) en la Guía para desarrolladores de [Managed Service para Apache Flink](#).

Para ver más ejemplos de consultas SQL de Flink Streaming, consulte [Consultas](#) en la [documentación de Apache Flink](#).

Limpiar su aplicación y los recursos dependientes

Elimine su bloc de notas de Studio

1. Abra la consola de Managed Service para Apache Flink
2. Elija MyNotebook.
3. Seleccione Acciones y luego Eliminar.

Elimine su base de datos y conexión AWS Glue

1. Abra la consola de AWS Glue en <https://console.aws.amazon.com/glue/>.
2. En la barra de navegación izquierda, elija Bases de datos. Marque la casilla de verificación situada junto a Predeterminado para seleccionarla. Seleccione Acción, Eliminar base de datos. Confirme la opción elegida.
3. En la barra de navegación de la izquierda, seleccione Conexiones. Marque la casilla de verificación situada junto a ZeppelinConnection para seleccionarla. Seleccione Acción, Eliminar conexión. Confirme la opción elegida.

Elimine la política y el rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el menú de navegación izquierdo, elija Roles.
3. Utilice la barra de búsqueda para buscar el rol de ZeppelinRole.
4. Seleccione el rol ZeppelinRole. Seleccione Eliminar rol. Confirme la eliminación.

Elimine su grupo de registro de CloudWatch

La consola crea un grupo de registro de CloudWatch y un flujo de registro cuando se crea la aplicación con la consola. No tiene un grupo de registros ni un flujo si creó la aplicación con AWS CLI.

1. Abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación izquierda, elija Grupos de registro.
3. Elija el grupo de registros /AWS/KinesisAnalytics/myNotebook.
4. Elija Acciones, Eliminar grupo(s) de registro(s). Confirme la eliminación.

Limpie los recursos de flujo de datos de Kinesis

Para eliminar la transmisión de Kinesis, abra la consola de flujo de datos de Kinesis, seleccione la transmisión de Kinesis y seleccione Acciones, Eliminar.

Limpie recursos MSK

Siga los pasos de esta sección si ha creado un clúster de Amazon MSK para este tutorial. Esta sección contiene instrucciones para limpiar la instancia de cliente de Amazon EC2, Amazon VPC y el clúster de Amazon MSK.

Elimine su clúster de Amazon MSK

Siga los pasos de esta sección si ha creado un clúster de Amazon MSK para este tutorial.

1. Abra la consola de Amazon MSK en <https://console.aws.amazon.com/msk/home?region=us-east-1#/home/>.
2. Elija AWSKafkaTutorialCluster. Elija Eliminar. Ingrese **delete** en la ventana que aparece y confirme su selección.

Terminar su instancia de cliente

Siga los pasos de esta sección si ha creado una instancia de cliente de Amazon EC2 para este tutorial.

1. Abra la consola de Amazon EC2 en <https://console.aws.amazon.com/ec2/>.
2. En el panel de navegación izquierdo, seleccione Instancias.

3. Seleccione la casilla de verificación situada junto a ZeppelinClient para seleccionarla.
4. Seleccione Estado de la instancia y Terminar instancia.

Elimine su VPC de Amazon

Siga los pasos de esta sección si ha creado un clúster de Amazon VPC para este tutorial.

1. Abra la consola de Amazon EC2 en <https://console.aws.amazon.com/ec2/>.
2. Seleccione Interfaces de red en la barra de navegación izquierda.
3. Escriba los términos de búsqueda en el cuadro de búsqueda y presione “Ingresar”.
4. Seleccione la casilla de verificación del encabezado de la tabla para seleccionar todas las interfaces de red mostradas.
5. Elija Acciones, Desasociar. En la ventana que aparece, seleccione Activar en Desprendimiento forzado. Seleccione Separar y espere a que todas las interfaces de red alcancen el estado Disponible.
6. Seleccione la casilla de verificación del encabezado de la tabla para seleccionar todas las interfaces de red mostradas.
7. Elija Acciones, Eliminar. Confirme la acción.
8. Abra la consola de Amazon VPC en <https://console.aws.amazon.com/vpc/>.
9. Seleccione AWSKafkaTutorialVPC. Elija Acciones, Eliminar VPC. Ingrese **delete** y confirme la eliminación.

Tutorial: Implementación como una aplicación con estado duradero

El siguiente tutorial muestra cómo implementar un bloc de notas de Studio como un servicio gestionado para una aplicación Apache Flink con estado duradero.

Este tutorial contiene las siguientes secciones:

- [Configuración](#)
- [Implemente una aplicación con un estado duradero mediante el AWS Management Console](#)
- [Implemente una aplicación con un estado duradero mediante el AWS CLI](#)

Configuración

Cree un nuevo bloc de notas de Studio siguiendo las [Tutorial sobre cómo crear un bloc de notas de Studio](#), utilizando flujo de datos Kinesis o Amazon MSK. Asigne un nombre al bloc de notas de Studio. ExampleTestDeploy

Implemente una aplicación con un estado duradero mediante el AWS Management Console

1. Agregue una ubicación de bucket S3 en la que desee almacenar el código empaquetado en la ubicación del código de la aplicación (opcional) en la consola. Esto permite seguir los pasos necesarios para implementar y ejecutar la aplicación directamente desde el bloc de notas.
2. Añada los permisos necesarios al rol de la aplicación para habilitar el rol que está utilizando para leer y escribir en un bucket de Amazon S3 y para lanzar una aplicación de servicio gestionado para Apache Flink:
 - AmazonS3FullAccess
 - Amazonmanaged-flinkFullAccess
 - Acceda a sus orígenes, destinos y VPC, según corresponda. Para obtener más información, consulte [Permisos de IAM para blocs de notas de Studio](#).
3. Consulte el siguiente código de muestra:

```
%flink.ssql(type=update)
CREATE TABLE exampleoutput (
  'ticket' VARCHAR,
  'price' DOUBLE
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'ExampleOutputStream',
  'aws.region' = 'us-east-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json'
);
```

```
INSERT INTO exampleoutput SELECT ticker, price FROM exampleinputstream
```

4. Al lanzar esta característica, verá un nuevo menú desplegable en la esquina superior derecha de cada nota de su bloc de notas con el nombre del bloc. Puede hacer lo siguiente:

- Consulte la configuración del bloc de notas de Studio en. AWS Management Console
- Cree su Zeppelin Note y expórtelo a Amazon S3. En ese momento, proporcione un nombre para la aplicación y seleccione Crear y exportar. Recibirá una notificación cuando se complete la exportación.
- Si lo necesita, puede ver y ejecutar cualquier prueba adicional en el ejecutable en Amazon S3.
- Una vez completada la compilación, podrá implementar su código como una aplicación de streaming de Kinesis con estado duradero y escalado automático.
- Utilice el menú desplegable y elija Deploy Zeppelin Note as Kinesis streaming application (Implementar Zeppelin Note como aplicación de streaming de Kinesis). Revise el nombre de la aplicación y elija Implementar mediante consola. AWS
- Esto lo llevará a la AWS Management Console página para crear un servicio gestionado para la aplicación Apache Flink. Tenga en cuenta que el nombre de la aplicación, el paralelismo, la ubicación del código, las funciones predeterminadas de Glue DB, VPC (si corresponde) y roles de IAM se han rellenado previamente. Compruebe que los roles de IAM tengan los permisos necesarios para acceder a sus orígenes y destinos. Las instantáneas están habilitadas de forma predeterminada para una administración duradera del estado de las aplicaciones.
- Elija Crear aplicación.
- Puede elegir configurar y modificar cualquier configuración y seleccionar Ejecutar para iniciar la aplicación de streaming.

Implemente una aplicación con un estado duradero mediante el AWS CLI

Para implementar una aplicación mediante el AWS CLI, debe actualizar su modelo de servicio AWS CLI para usar el que se proporciona con la información de la versión beta 2. Para obtener información acerca de cómo utilizar el modelo de servicio actualizado, consulte [Configuración](#).

El siguiente ejemplo de código inicia un bloc de notas de Studio:

```
aws kinesisanalyticsv2 create-application \  
  --application-name <app-name> \  
  --runtime-environment ZEPPELIN-FLINK-3_0 \  
  --application-mode INTERACTIVE \  
  --service-execution-role <iam-role> \  
  --application-configuration '{  
    "ZeppelinApplicationConfiguration": {  
      "CatalogConfiguration": {
```

```

    "GlueDataCatalogConfiguration": {
      "DatabaseARN": "arn:aws:glue:us-east-1:<account>:database/<glue-database-
name>"
    }
  },
  "FlinkApplicationConfiguration": {
    "ParallelismConfiguration": {
      "ConfigurationType": "CUSTOM",
      "Parallelism": 4,
      "ParallelismPerKPU": 4
    }
  },
  "DeployAsApplicationConfiguration": {
    "S3ContentLocation": {
      "BucketARN": "arn:aws:s3:::<s3bucket>",
      "BasePath": "/something/"
    }
  },
  "VpcConfigurations": [
    {
      "SecurityGroupIds": [
        "<security-group>"
      ],
      "SubnetIds": [
        "<subnet-1>",
        "<subnet-2>"
      ]
    }
  ]
} \
--region us-east-1

```

El siguiente ejemplo de código inicia un bloc de notas de Studio:

```

aws kinesisanalyticv2 start-application \
  --application-name <app-name> \
  --region us-east-1 \
  --no-verify-ssl

```

El código siguiente devuelve la URL de la página del bloc de notas de Apache Zeppelin de una aplicación:

```
aws kinesisanalyticstv2 create-application-presigned-url \  
  --application-name <app-name> \  
  --url-type ZEPPELIN_UI_URL \  
  
  --region us-east-1 \  
  --no-verify-ssl
```

Ejemplos

Los siguientes ejemplos de consultas muestran cómo analizar los datos mediante consultas de ventana en un bloc de notas de Studio.

- [Creación de tablas con Amazon MSK/Apache Kafka](#)
- [Creación de tablas con Kinesis](#)
- [Ventana de saltos de tamaño constante](#)
- [Ventana deslizante](#)
- [SQL interactivo](#)
- [BlackHole Conector SQL](#)
- [Generador de datos](#)
- [Scala interactiva](#)
- [Python interactivo](#)
- [Python, SQL y Scala interactivos](#)
- [Flujo de datos de Kinesis entre cuentas](#)

Para obtener información sobre la configuración de las consultas SQL de Apache Flink, consulte [Flink en blocs de notas de Zeppelin](#) para análisis de datos interactivos.

Para ver su aplicación en el panel de control de Apache Flink, elija TRABAJO FLINK en la página Bloc de notas de Zeppelin de su aplicación.

Para obtener más información sobre las consultas de ventanas, consulte [Ventanas](#) en la [Documentación de Apache Flink](#).

Para ver más ejemplos de consultas SQL de Apache Flink, consulte [Consultas](#) en la [Documentación de Apache Flink](#).

Creación de tablas con Amazon MSK/Apache Kafka

Puede utilizar el conector Amazon MSK Flink con Managed Service para Apache Flink Studio para autenticar su conexión con la autenticación Plaintext, SSL o IAM. Cree sus tablas con las propiedades específicas según sus requisitos.

```
-- Plaintext connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- SSL connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SSL',
  'properties.ssl.truststore.location' = '/usr/lib/jvm/java-11-amazon-corretto/lib/
security/cacerts',
  'properties.ssl.truststore.password' = 'changeit',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- IAM connection (or for MSK Serverless)

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
```

```
'connector' = 'kafka',
'topic' = 'your_topic',
'properties.bootstrap.servers' = '<bootstrap servers>',
'properties.security.protocol' = 'SASL_SSL',
'properties.sasl.mechanism' = 'AWS_MSK_IAM',
'properties.sasl.jaas.config' = 'software.amazon.msk.auth.iam.IAMLoginModule
required;',
'properties.sasl.client.callback.handler.class' =
'software.amazon.msk.auth.iam.IAMClientCallbackHandler',
'properties.group.id' = 'myGroup',
'scan.startup.mode' = 'earliest-offset',
'format' = 'json'
);
```

Puede combinarlas con otras propiedades en [Apache Kafka SQL Connector](#).

Creación de tablas con Kinesis

En el siguiente ejemplo, se crea una tabla con Kinesis:

```
CREATE TABLE KinesisTable (
  `column1` BIGINT,
  `column2` BIGINT,
  `column3` BIGINT,
  `column4` STRING,
  `ts` TIMESTAMP(3)
)
PARTITIONED BY (column1, column2)
WITH (
  'connector' = 'kinesis',
  'stream' = 'test_stream',
  'aws.region' = '<region>',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'csv'
);
```

Para obtener más información acerca de otras propiedades que puede usar, consulte [Conector SQL de Amazon Kinesis Data Streams](#).

Ventana de saltos de tamaño constante

La siguiente consulta SQL de Flink Streaming selecciona de la tabla `ZeppelinTopic` el precio más alto de cada intervalo de cinco segundos:

```
%flink.ssql(type=update)
SELECT TUMBLE_END(event_time, INTERVAL '5' SECOND) as winend, MAX(price) as
  five_second_high, ticker
FROM ZeppelinTopic
GROUP BY ticker, TUMBLE(event_time, INTERVAL '5' SECOND)
```

Ventana deslizante

La siguiente consulta SQL de Apache Flink Streaming selecciona de la tabla `ZeppelinTopic` el precio más alto de cada ventana deslizante de cinco segundos:

```
%flink.ssql(type=update)
SELECT HOP_END(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND) AS winend,
  MAX(price) AS sliding_five_second_max
FROM ZeppelinTopic//or your table name in AWS Glue
GROUP BY HOP(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND)
```

SQL interactivo

En este ejemplo, se imprime el tiempo máximo del evento y el tiempo de procesamiento y la suma de los valores de la tabla de valores clave. Asegúrese de tener el ejemplo del script de generación de datos de [the section called “Generador de datos”](#) en ejecución. Para probar otras consultas SQL, como el filtrado y las uniones, en su bloc de notas de Studio, consulte la documentación de Apache Flink: [Consultas](#) en la documentación de Apache Flink.

```
%flink.ssql(type=single, parallelism=4, refreshInterval=1000, template=<h1>{2}</h1>
  records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints how many records from the `key-value-stream` we have
  seen so far, along with the current processing and event time.
SELECT
  MAX(`et`) as `et`,
  MAX(`pt`) as `pt`,
  SUM(`value`) as `sum`
FROM
  `key-values`
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive tumbling window query that displays the number of records observed
  per (event time) second.
```

```
-- Browse through the chart views to see different visualizations of the streaming
result.
SELECT
  TUMBLE_START(`et`, INTERVAL '1' SECONDS) as `window`,
  `key`,
  SUM(`value`) as `sum`
FROM
  `key-values`
GROUP BY
  TUMBLE(`et`, INTERVAL '1' SECONDS),
  `key`;
```

BlackHole Conector SQL

El conector BlackHole SQL no requiere que cree una transmisión de datos de Kinesis o un clúster de Amazon MSK para probar sus consultas. Para obtener información sobre el conector BlackHole SQL, consulte el [conector BlackHole SQL](#) en la documentación de Apache Flink. En este ejemplo, el catálogo predeterminado es un catálogo en memoria.

```
%flink.ssql

CREATE TABLE default_catalog.default_database.blackhole_table (
  `key` BIGINT,
  `value` BIGINT,
  `et` TIMESTAMP(3)
) WITH (
  'connector' = 'blackhole'
)
```

```
%flink.ssql(parallelism=1)

INSERT INTO `test-target`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-source`
WHERE
  `key` > 3
```

```
%flink.ssql(parallelism=2)

INSERT INTO `default_catalog`.`default_database`.`blackhole_table`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-target`
WHERE
  `key` > 7
```

Generador de datos

En este ejemplo, se utiliza Scala para generar datos de muestra. Puede utilizar estos datos de ejemplo para probar varias consultas. Utilice la instrucción crear tabla para crear la tabla de valores clave.

```
import org.apache.flink.streaming.api.functions.source.datagen.DataGeneratorSource
import org.apache.flink.streaming.api.functions.source.datagen.RandomGenerator
import org.apache.flink.streaming.api.scala.DataStream

import java.sql.Timestamp

// ad-hoc convenience methods to be defined on Table
implicit class TableOps[T](table: DataStream[T]) {
  def asView(name: String): DataStream[T] = {
    if (stenv.listTemporaryViews.contains(name)) {
      stenv.dropTemporaryView("`" + name + "`")
    }
    stenv.createTemporaryView("`" + name + "`", table)
    return table;
  }
}
```

```
%flink(parallelism=4)
val stream = senv
  .addSource(new DataGeneratorSource(RandomGenerator.intGenerator(1, 10), 1000))
  .map(key => (key, 1, new Timestamp(System.currentTimeMillis)))
  .asView("key-values-data-generator")
```



```
%flink.ssql(parallelism=4)
-- no need to define the paragraph type with explicit parallelism (such as
"%flink.ssql(parallelism=2)")
-- in this case the INSERT query will inherit the parallelism of the of the above
paragraph
INSERT INTO `key-values`
SELECT
  `_1` as `key`,
  `_2` as `value`,
  `_3` as `et`
FROM
  `key-values-data-generator`
```

Scala interactiva

Esta es la traducción en Scala del [the section called “SQL interactivo”](#). Para ver más ejemplos de Scala, consulte la [API de tablas](#) en la documentación de Apache Flink.

```
%flink
import org.apache.flink.api.scala._
import org.apache.flink.table.api._
import org.apache.flink.table.api.bridge.scala._

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
  def asView(name: String): Table = {
    if (stenv.listTemporaryViews.contains(name)) {
      stenv.dropTemporaryView(name)
    }
    stenv.createTemporaryView(name, table)
    return table;
  }
}
```

```
%flink(parallelism=4)

// A view that computes many records from the `key-values` we have seen so far, along
with the current processing and event time.
val query01 = stenv
  .from("`key-values`")
  .select(
    $"et".max().as("et"),
```

```

    $"pt".max().as("pt"),
    $"value".sum().as("sum")
  ).asView("query01")

```

```

%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

```

```

-- An interactive query prints the query01 output.
SELECT * FROM query01

```

```

%flink(parallelism=4)

```

```

// An tumbling window view that displays the number of records observed per (event
time) second.

```

```

val query02 = stenv
  .from("`key-values`")
  .window(Tumble over 1.seconds on $"et" as $"w")
  .groupBy($"w", $"key")
  .select(
    $"w".start.as("window"),
    $"key",
    $"value".sum().as("sum")
  ).asView("query02")

```

```

%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

```

```

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming
result.
SELECT * FROM `query02`

```

Python interactivo

Esta es la traducción en Python del [the section called “SQL interactivo”](#). Para ver más ejemplos de Python, consulte [la API de tablas](#) en la documentación de Apache Flink.

```

%flink.pyflink
from pyflink.table.table import Table

def as_view(table, name):
    if (name in st_env.list_temporary_views()):

```

```

    st_env.drop_temporary_view(name)
    st_env.create_temporary_view(name, table)
    return table

```

```
Table.as_view = as_view
```

```
%flink.pyflink(parallelism=16)
```

```
# A view that computes many records from the `key-values` we have seen so far, along
with the current processing and event time
```

```

st_env \
  .from_path("`keyvalues`") \
  .select(", ".join([
    "max(et) as et",
    "max(pt) as pt",
    "sum(value) as sum"
  ])) \
  .as_view("query01")

```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)
```

```
-- An interactive query prints the query01 output.
```

```
SELECT * FROM query01
```

```
%flink.pyflink(parallelism=16)
```

```
# A view that computes many records from the `key-values` we have seen so far, along
with the current processing and event time
```

```

st_env \
  .from_path("`key-values`") \
  .window(Tumble.over("1.seconds").on("et").alias("w")) \
  .group_by("w, key") \
  .select(", ".join([
    "w.start as window",
    "key",
    "sum(value) as sum"
  ])) \
  .as_view("query02")

```

```
%flink.ssql(type=update, parallelism=16, refreshInterval=1000)
```

```
-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming
  result.
SELECT * FROM `query02`
```

Python, SQL y Scala interactivos

Puede utilizar cualquier combinación de SQL, Python y Scala en su bloc de notas para el análisis interactivo. En un bloc de notas de Studio que vaya a implementar como una aplicación con un estado duradero, puede usar una combinación de SQL y Scala. En este ejemplo, se muestran las secciones que se ignoran y las que se implementan en la aplicación con un estado duradero.

```
%flink.sql
CREATE TABLE `default_catalog`.`default_database`.`my-test-source` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-source-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink.sql
CREATE TABLE `default_catalog`.`default_database`.`my-test-target` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-target-stream',
  'aws.region' = 'eu-west-1',
```

```
'scan.stream.initpos' = 'LATEST',
'format' = 'json',
'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink()

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
  def asView(name: String): Table = {
    if (stenv.listTemporaryViews.contains(name)) {
      stenv.dropTemporaryView(name)
    }
    stenv.createTemporaryView(name, table)
    return table;
  }
}
```

```
%flink(parallelism=1)
val table = stenv
  .from("`default_catalog`.`default_database`.`my-test-source`")
  .select($"key", $"value", $"et")
  .filter($"key" > 10)
  .asView("query01")
```

```
%flink.ssql(parallelism=1)

-- forward data
INSERT INTO `default_catalog`.`default_database`.`my-test-target`
SELECT * FROM `query01`
```

```
%flink.ssql(type=update, parallelism=1, refreshInterval=1000)

-- forward data to local stream (ignored when deployed as application)
SELECT * FROM `query01`
```

```
%flink

// tell me the meaning of life (ignored when deployed as application!)
print("42!")
```

Flujo de datos de Kinesis entre cuentas

Para usar un flujo de datos de Kinesis que esté en una cuenta distinta de la cuenta que tiene su bloc de notas de Studio, cree un rol de ejecución de servicios en la cuenta en la que se ejecuta el bloc de notas de Studio y una política de confianza de roles en la cuenta que tiene el flujo de datos. Utilice `aws.credentials.provider`, `aws.credentials.role.arn`, y `aws.credentials.role.sessionName` en el conector de Kinesis de la instrucción DDL de creación de tabla para crear una tabla con el flujo de datos.

Utilice la siguiente función de ejecución de servicios para la cuenta de blocs de notas de Studio.

```
{
  "Sid": "AllowNotebookToAssumeRole",
  "Effect": "Allow",
  "Action": "sts:AssumeRole"
  "Resource": "*"
}
```

Utilice la política `AmazonKinesisFullAccess` y la siguiente política de confianza de roles para la cuenta de flujo de datos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<accountID>:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

Utilice el siguiente párrafo para la declaración de creación de la tabla.

```
%flink.sql
CREATE TABLE test1 (
  name VARCHAR,
  age BIGINT
```

```
) WITH (  
'connector' = 'kinesis',  
'stream' = 'stream-assume-role-test',  
'aws.region' = 'us-east-1',  
'aws.credentials.provider' = 'ASSUME_ROLE',  
'aws.credentials.role.arn' = 'arn:aws:iam::<accountID>:role/stream-assume-role-test-  
role',  
'aws.credentials.role.sessionName' = 'stream-assume-role-test-session',  
'scan.stream.initpos' = 'TRIM_HORIZON',  
'format' = 'json'  
)
```

Solución de problemas

Esta sección contiene información sobre la solución de problemas de los blocs de notas de Studio.

Detener una aplicación atascada

Para detener una aplicación que está atascada en un estado transitorio, llame a la [StopApplication](#) acción con el Force parámetro establecido en true. Para obtener más información, consulte [Aplicaciones en ejecución](#) en la [Guía para desarrolladores de Managed Service para Apache Flink](#).

Implementación como una aplicación con un estado duradero en una VPC sin acceso a Internet

La deploy-as-application función Managed Service for Apache Flink Studio no admite aplicaciones de VPC sin acceso a Internet. Le recomendamos que cree la aplicación en Studio y, a continuación, utilice Managed Service para Apache Flink para crear manualmente una aplicación de Flink y seleccionar el archivo zip que creó en su bloc de notas.

Los pasos siguientes describen este abordaje:

1. Compile y exporte su aplicación de Studio a Amazon S3. Debe ser un archivo zip.
2. Cree una aplicación de Managed Service para Apache Flink manualmente con una ruta de código que haga referencia a la ubicación del archivo zip en Amazon S3. Además, necesitará configurar la aplicación con las siguientes variables env (2 groupID, 3 var en total):
3. `kinesis.analytics.flink.run.options`

- a. python: source/note.py
 - b. Archivo jar: lib/ .jar PythonApplicationDependencies
4. managed.deploy_as_app.options
- DatabaseARN: *<glue database ARN (Amazon Resource Name)>*
5. Es posible que necesite conceder permisos a las funciones de IAM de Managed Service para Apache Flink Studio y de Managed Service para Apache Flink para los servicios que su aplicación utilice. Puede usar el mismo rol de IAM para ambas aplicaciones.

Reducción del deploy-as-app tamaño D y del tiempo de construcción

Las aplicaciones de Studio deploy-as-app for Python empaquetan todo lo que está disponible en el entorno Python porque no podemos determinar qué bibliotecas necesita. Esto puede resultar en un tamaño mayor del necesario. deploy-as-app El siguiente procedimiento muestra cómo reducir el tamaño de la aplicación deploy-as-app Python mediante la desinstalación de las dependencias.

Si está creando una aplicación de Python con una deploy-as-app función de Studio, podría considerar eliminar los paquetes de Python preinstalados del sistema si sus aplicaciones no dependen de ella. Esto no solo ayudará a reducir el tamaño final del artefacto para evitar sobrepasar el límite de tamaño de las aplicaciones por servicio, sino que también mejorará el tiempo de compilación de las aplicaciones que cuenten con esta función. deploy-as-app

Puede ejecutar el siguiente comando para enumerar todos los paquetes de Python instalados con sus respectivos tamaños de instalación y eliminar selectivamente los que tienen un tamaño significativo.

```
%flink.pyflink

!pip list --format freeze | awk -F = {'print $1'} | xargs pip show | grep -E
'Location:|Name:' | cut -d ' ' -f 2 | paste -d ' ' - - | awk '{gsub("-", "_", $1); print
$2 "/" tolower($1)}' | xargs du -sh 2> /dev/null | sort -hr
```

Note

apache-beam es necesario para que Flink Python funcione. Nunca debe eliminar este paquete ni sus dependencias.

A continuación se encuentra la lista de paquetes de Python preinstalados en Studio V2 que se pueden eliminar:

```
scipy
statsmodels
plotnine
seaborn
llvmlite
bokeh
pandas
matplotlib
botocore
boto3
numba
```

Para eliminar un paquete de Python del bloc de notas de Zeppelin:

1. Compruebe si su aplicación depende del paquete, o de alguno de los paquetes que lo utilice, antes de eliminarla. Puede identificar los dependientes de un paquete mediante [pipdeptree](#).
2. Ejecución del siguiente comando para eliminar un paquete:

```
%flink.pyflink
!pip uninstall -y <package-to-remove>
```

3. Si necesita recuperar un paquete que se eliminó por error, ejecute el siguiente comando:

```
%flink.pyflink
!pip install <package-to-install>
```

Example Ejemplo: Elimine el **scipy** paquete antes de implementar su aplicación Python con la `deploy-as-app` función.

1. Use `pipdeptree` para descubrir todos los consumidores de `scipy` y comprobar si puede eliminar `scipy` de forma segura.
 - Instale la herramienta a través del bloc de notas:

```
%flink.pyflink
!pip install pipdeptree
```

- Obtenga un árbol de dependencias invertido de `scipy` al ejecutar:

```
%flink.pyflink
!pip -r -p scipy
```

Debería ver una salida similar a la siguiente (está condensada para mayor rapidez):

```
...
-----
scipy==1.8.0
### plotnine==0.5.1 [requires: scipy>=1.0.0]
### seaborn==0.9.0 [requires: scipy>=0.14.0]
### statsmodels==0.12.2 [requires: scipy>=1.1]
    ### plotnine==0.5.1 [requires: statsmodels>=0.8.0]
```

2. Inspeccione cuidadosamente el uso de `seaborn`, `statsmodels` y `plotnine` en sus aplicaciones. Si sus aplicaciones no dependen de ninguno de los paquetes `scipy`, `seaborn`, `statemodels` o `plotnine`, puede eliminarlos todos o solo los que sus aplicaciones no necesiten.
3. Elimine el paquete al ejecutar:

```
!pip uninstall -y scipy plotnine seaborn statemodels
```

Cancelación de trabajos

En esta sección, se muestra cómo cancelar los trabajos de Apache Flink a los que no puede acceder desde Apache Zeppelin. Si desea cancelar un trabajo de este tipo, vaya al panel de control de Apache Flink, copie el identificador del trabajo y utilícelo en uno de los siguientes ejemplos.

Para cancelar un solo trabajo:

```
%flink.pyflink
import requests

requests.patch("https://zeppelin-flink:8082/jobs/[job_id]", verify=False)
```

Para cancelar todos los trabajos en ejecución:

```
%flink.pyflink
```

```
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    if (job["status"] == "RUNNING"):
        print(requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
            verify=False))
```

Para cancelar todos los trabajos:

```
%flink.pyflink
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
        verify=False)
```

Reiniciar el intérprete Apache Flink

Para reiniciar el intérprete Apache Flink en su bloc de notas de Studio

1. Elija Configuración cerca de la esquina superior derecha de la pantalla.
2. Elija Intérprete.
3. Seleccione reiniciar y, a continuación, Aceptar.

Apéndice: Creación de políticas de IAM personalizadas

Normalmente se utilizan políticas de IAM gestionadas para permitir que su aplicación acceda a recursos dependientes. Si necesita un control más preciso sobre los permisos de la aplicación, puede utilizar una política de IAM personalizada. Esta sección contiene ejemplos de políticas de IAM personalizadas.

Note

En los siguientes ejemplos de políticas, sustituya el texto del marcador de posición por los valores de la aplicación.

Este tema contiene las siguientes secciones:

- [AWS Glue](#)
- [CloudWatch Registros](#)
- [Flujos de Kinesis](#)
- [Clústeres de Amazon MSK](#)

AWS Glue

La siguiente política de ejemplo concede permisos para obtener acceso a la base de datos de AWS Glue.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GlueTable",
      "Effect": "Allow",
      "Action": [
        "glue:GetConnection",
        "glue:GetTable",
        "glue:GetTables",
        "glue:GetDatabase",
        "glue:CreateTable",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<accountId>:connection/*",
        "arn:aws:glue:<region>:<accountId>:table/<database-name>/*",
        "arn:aws:glue:<region>:<accountId>:database/<database-name>",
        "arn:aws:glue:<region>:<accountId>:database/hive",
        "arn:aws:glue:<region>:<accountId>:catalog"
      ]
    }
  ],
}
```

```
{
  {
    "Sid": "GlueDatabase",
    "Effect": "Allow",
    "Action": "glue:GetDatabases",
    "Resource": "*"
  }
}
```

CloudWatch Registros

La siguiente política otorga permisos para acceder a CloudWatch los registros:

```
{
  {
    "Sid": "ListCloudwatchLogGroups",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:<region>:<accountId>:log-group:*"
    ]
  },
  {
    "Sid": "ListCloudwatchLogStreams",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "<LogGroupArn>:log-stream:*"
    ]
  },
  {
    "Sid": "PutCloudwatchLogs",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "<LogStreamArn>"
    ]
  }
}
```

Note

Si crea la aplicación mediante la consola, esta añadirá las políticas necesarias para acceder a CloudWatch los registros a su función de aplicación.

Flujos de Kinesis

Su aplicación puede usar un flujo de Kinesis como origen o destino. Su aplicación necesita permisos de lectura para leer desde un flujo de origen y permisos de escritura para escribir en un flujo de destino.

La siguiente política concede permisos para leer desde un flujo de Kinesis utilizado como fuente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisShardDiscovery",
      "Effect": "Allow",
      "Action": "kinesis:ListShards",
      "Resource": "*"
    },
    {
      "Sid": "KinesisShardConsumption",
      "Effect": "Allow",
      "Action": [
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:DescribeStream",
        "kinesis:DescribeStreamSummary",
        "kinesis:RegisterStreamConsumer",
        "kinesis:DeregisterStreamConsumer"
      ],
      "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
    },
    {
      "Sid": "KinesisEfoConsumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "arn:aws:kinesis:<region>:<account>:stream/<stream-name>/consumer/*"
  }
]
}

```

La siguiente política concede permisos para escribir en una transmisión de Kinesis utilizada como origen:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kinesis:PutRecord",
        "kinesis:PutRecords",
        "kinesis:DescribeStreamSummary",
        "kinesis:DescribeStream"
      ],
      "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
    }
  ]
}

```

Si su aplicación accede a un flujo de Kinesis cifrado, debe conceder permisos adicionales para acceder al flujo y a la clave de cifrado del flujo.

La siguiente política concede permisos para acceder a un flujo de origen cifrado y a la clave de cifrado del flujo:

```

{
  "Sid": "ReadEncryptedKinesisStreamSource",
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": [
    "<inputStreamKeyArn>"
  ]
}

```

La siguiente política concede permisos para acceder a un flujo de destino cifrado y a la clave de cifrado del flujo:

```
{
  "Sid": "WriteEncryptedKinesisStreamSink",
  "Effect": "Allow",
  "Action": [
    "kms:GenerateDataKey"
  ],
  "Resource": [
    "<outputStreamKeyArn>"
  ]
}
```

Clústeres de Amazon MSK

Para conceder acceso a un clúster de Amazon MSK, debe conceder acceso a la VPC del clúster. Para ver ejemplos de políticas de acceso a una VPC de Amazon, consulte [Permisos de aplicaciones de VPC](#).

Introducción a Amazon Managed Service para Apache Flink (DataStream API)

En esta sección, se presentan los conceptos fundamentales del servicio gestionado para Apache Flink y la DataStream API. Describe las opciones disponibles para crear y probar sus aplicaciones. También proporciona instrucciones para instalar las herramientas necesarias para completar los tutoriales de esta guía y crear su primera aplicación.

Temas

- [Componentes de la aplicación de Managed Service para Apache Flink](#)
- [Requisitos previos para completar los ejercicios](#)
- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)
- [Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)
- [Paso 4: Eliminar recursos de AWS](#)
- [Paso 5: Siguiendo los pasos](#)

Componentes de la aplicación de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Java/Apache Maven o Scala que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

Una aplicación de Managed Service para Apache Flink tiene los siguientes componentes:

- **Propiedades de tiempo de ejecución:** puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- **Fuente:** la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un bucket de Amazon S3, etc. Para obtener más información, consulte [Orígenes](#).
- **Operadores:** la aplicación procesa los datos mediante uno o más operadores. Un operador puede transformar, enriquecer o agregar datos. Para obtener más información, consulte [Operadores de la API de DataStream](#).

- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores](#).

Después de crear, compilar y empaquetar el código de la aplicación, debe cargar el paquete del código a un bucket de Amazon Simple Storage Service (Amazon S3). Luego debe crear la aplicación de Managed Service para Apache Flink. Introduzca la ubicación del paquete del código, un flujo de datos de Kinesis como origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos para completar los ejercicios

Para completar los pasos de esta guía, debe disponer de lo siguiente:

- [Java Development Kit \(JDK\), versión 11](#). Establezca la variable de entorno JAVA_HOME para señalar la ubicación de la instalación del JDK.
- Le recomendamos utilizar un entorno de desarrollo (como [Eclipse Java Neon](#) o [IntelliJ Idea](#)) para desarrollar y compilar su aplicación.
- [Cliente Git](#). Si aún no lo ha hecho, instale el cliente Git.
- [Apache Maven Compiler Plugin](#). Maven debe estar en su ruta de trabajo. Para probar la instalación de Apache Maven, introduzca lo siguiente:

```
$ mvn -version
```

Para empezar, vaya a [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#).

Paso 1: Configurar una cuenta de AWS y crear un usuario administrador

Antes de utilizar Managed Service para Apache Flink por primera vez, complete las siguientes tareas:

Registrarse en una Cuenta de AWS

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

Cómo registrarse para una Cuenta de AWS

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

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

Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, [asigne acceso administrativo a un usuario administrativo](#) y utilice únicamente el usuario raíz para realizar [tareas que requieran acceso de usuario raíz](#).

AWS le enviará un correo electrónico de confirmación luego de completar el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en <https://aws.amazon.com/> y seleccionando Mi cuenta.

Crear un usuario administrativo

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

Protección de su Usuario raíz de la cuenta de AWS

1. Inicie sesión en la [AWS Management Console](#) como propietario de cuenta, elija Usuario raíz e ingrese el email de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Signing in as the root user](#) en la Guía del usuario de AWS Sign-In.

2. Active la autenticación multifactor (MFA) para el usuario raíz.

Para obtener instrucciones, consulte [Habilitar un dispositivo MFA virtual para el usuario raíz de la Cuenta de AWS \(consola\)](#) en la Guía del usuario de IAM.

Creación de un usuario administrativo

1. Activar IAM Identity Center

Para conocer las instrucciones, consulte [Habilitar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center.

2. En IAM Identity Center, otorga acceso administrativo a un usuario administrativo.

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

Cómo iniciar sesión como usuario administrativo

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

Para obtener ayuda para iniciar sesión con un usuario del IAM Identity Center, consulte [Iniciar sesión en el portal de acceso de AWS](#) en la Guía del Usuario de AWS Sign-In.

Conceder acceso programático

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

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

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • En AWS CLI, consulte Configuración de la AWS CLI para usar AWS IAM Identity Center en la Guía del usuario de AWS Command Line Interface.

¿Qué usuario necesita acceso programático?	Para	Mediante
		<ul style="list-style-type: none">• En los SDK de AWS, las herramientas y las API de AWS, consulte Autenticación de IAM Identity Center en la Guía de referencia de SDK de AWS y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del Usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para los SDK de AWS y las herramientas, consulte Autenticación mediante credenciales a largo plazo en la Guía de referencia de SDK de AWS y herramientas. • Para las API de AWS, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Paso siguiente

[Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)

Paso 2: Configuración de la AWS Command Line Interface (AWS CLI)

En este paso, debe descargar y configurar la AWS CLI para usarla con Managed Service para Apache Flink.

Note

En los ejercicios introductorios de esta guía se presupone que está utilizando las credenciales de administrador (`adminuser`) en su cuenta para realizar las operaciones.

Note

Si ya tiene la AWS CLI instalada, es posible que tenga que actualizarla para obtener las últimas funcionalidades. Para obtener más información, consulte [Instalar la AWS Command Line Interface](#) en la Guía del usuario de AWS Command Line Interface. Ejecute el siguiente comando para comprobar la versión de la AWS CLI:

```
aws --version
```

Los ejercicios de este tutorial requieren la siguiente versión de la AWS CLI o posterior:

```
aws-cli/1.16.63
```

Para configurar la AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de la AWS Command Line Interface:
 - [Instalación de la AWS Command Line Interface](#)
 - [Configuración de la AWS CLI](#)
2. Añada un perfil con nombre para el usuario administrador en el archivo `config` de la AWS CLI. Puede utilizar este perfil cuando ejecute los comandos de la AWS CLI. Para obtener más información sobre los perfiles con nombre, consulte [Perfiles con nombre](#) en la Guía del usuario de AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Para ver una lista de las regiones de AWS disponibles, consulte [Regiones y puntos de conexión](#) en la Referencia general de Amazon Web Services.

Note

El código y los comandos de ejemplo de este tutorial utilizan la región Oeste de EE. UU. (Oregón). Para usar una región diferente, cambie la región en el código y los comandos de este tutorial por la región que desea usar.

3. Verifique la configuración introduciendo el siguiente comando de ayuda en el símbolo del sistema:

```
aws help
```

Tras configurar una AWS cuentaAWS CLI, puede realizar el siguiente ejercicio, en el que configurará una aplicación de ejemplo y probará la end-to-end configuración.

Paso siguiente

[Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)

Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink

En este ejercicio, deberá crear una aplicación de Managed Service para Apache Flink con flujos de datos como origen y receptor.

Esta sección contiene los siguientes pasos:

- [Crear dos Amazon Kinesis Data Streams](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Paso siguiente](#)

Crear dos Amazon Kinesis Data Streams

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, cree dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`). Su aplicación utiliza estos flujos para los flujos de origen y destino de la aplicación.

Puede crear estos flujos mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para crear flujos de datos (AWS CLI)

1. Para crear el primer flujo (`ExampleInputStream`), utilice el siguiente comando de la AWS CLI `create-stream` de Amazon Kinesis.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

2. Para crear el segundo flujo que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a `ExampleOutputStream`.

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Más adelante en el tutorial, se ejecuta el script `stock.py` para enviar datos a la aplicación.

```
$ python stock.py
```

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo [Project Object Model \(pom.xml\)](#) contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.java` contiene el método `main` que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos usando un objeto `StreamExecutionEnvironment`.
- La aplicación crea conectores de origen y recepción mediante propiedades estáticas. Para utilizar propiedades dinámicas de la aplicación, utilice los métodos `createSourceFromApplicationProperties` y `createSinkFromApplicationProperties` para crear los conectores. Estos métodos leen las propiedades de la aplicación para configurar los conectores.

Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar el código de la aplicación

En esta sección, se utiliza el compilador Apache Maven para crear el código de Java para la aplicación. Para obtener más información sobre la instalación de Apache Maven y el Java Development Kit (JDK), consulte [Requisitos previos para completar los ejercicios](#).

Para compilar el código de la aplicación

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar el código de una de las dos formas siguientes:

- Utilice la herramienta de línea de comandos de Maven. Cree su archivo JAR ejecutando el siguiente comando en el directorio que contiene el archivo `pom.xml`:

```
mvn package -Dflink.version=1.15.3
```

- Use el entorno de desarrollo. Consulte la documentación de su entorno de desarrollo para obtener más información.

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Puede cargar el paquete como un archivo JAR o puede comprimir el paquete y cargarlo como un archivo ZIP. Si crea su aplicación mediante la AWS CLI, debe especificar el tipo de contenido del código (JAR o ZIP).

2. Si hay errores al compilar, verifique que la variable de entorno `JAVA_HOME` se ha configurado correctamente.

Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Cargar el código de Java de streaming de Apache Flink

En esta sección, creará un bucket de Amazon Simple Storage Service (Amazon S3) y cargará el código de la aplicación.

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket.

3. Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. En la consola de Amazon S3, elija el *<username>*bucket ka-app-code- y elija Upload.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior. Elija Siguiente.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Puede crear y ejecutar una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI.

Note

Cuando crea la aplicación mediante la consola, sus recursos AWS Identity and Access Management (de IAM) y de Amazon CloudWatch Logs se crean automáticamente. Si crea la aplicación mediante la AWS CLI, debe crear estos recursos por separado.

Temas

- [Crear y ejecutar la aplicación \(consola\)](#)
- [Crear y ejecutar la aplicación \(AWS CLI\)](#)

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.

4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (*012345678901*) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": [
```

```

        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
},
{
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
},
{
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
}
]
}

```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, en ID de grupo, escriba **ProducerConfigProperties**.
5. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

6. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
7. Para el CloudWatch registro, active la casilla Activar.
8. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: /aws/kinesis-analytics/MyApplication
- Flujo de registro: kinesis-analytics-log-stream

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

En la MyApplication página, selecciona Detener. Confirme la acción.

Actualizar la aplicación

Mediante la consola, puede actualizar la configuración de la aplicación, tal como sus propiedades, ajustes de monitorización y la ubicación o el nombre de archivo JAR de la aplicación. También puede volver a cargar el JAR de la aplicación del bucket de Amazon S3 si necesita actualizar el código de la aplicación.

En la MyApplication página, elija Configurar. Actualice la configuración de la aplicación y elija Actualizar.

Crear y ejecutar la aplicación (AWS CLI)

En esta sección, se utiliza AWS CLI para crear y ejecutar la aplicación Managed Service para Apache Flink. Managed Service para Apache Flink utiliza el comando de la AWS CLI de `kinesisanalyticsv2` para crear e interactuar con las aplicaciones de Managed Service para Apache Flink.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción `read` en el flujo de origen y otra que concede permisos para las acciones `write` en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

`AKReadSourceStreamWriteSinkStream`. Reemplace *username* por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Sustituya el ID de la cuenta en los nombres de recurso de Amazon (ARN) (*012345678901*) por el ID de su cuenta.

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

```

"Statement": [
  {
    "Sid": "S3",
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "Resource": ["arn:aws:s3:::ka-app-code-username",
      "arn:aws:s3:::ka-app-code-username/*"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial sobre cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Note

Para acceder a otros servicios de Amazon, puede usar AWS SDK for Java. Managed Service para Apache Flink establece automáticamente las credenciales requeridas por el SDK con las del rol de IAM de ejecución del servicio asociada a su aplicación. No hace falta realizar ningún otro paso.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Roles, Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS. En Elegir el servicio que usará este rol, elija Kinesis. En Seleccionar su caso de uso, elija Kinesis Analytics.

Elija Siguiente: permisos.

4. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
5. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos del rol.

6. Asocie la política de permisos al rol.

Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Asocie la política que ha creado en el paso anterior, [the section called “Crear una política de permisos”](#).

- a. En la página Resumen, elija la pestaña Permisos.

- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la ReadSourceStreamWriteSinkStream política AK y elija Adjuntar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación de Managed Service para Apache Flink

1. Copie el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (*username*) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de ejemplo (*012345678901*) del rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "flink.stream.initpos": "LATEST",
            "aws.region": "us-west-2",
            "AggregationEnabled": "false"
          }
        }
      ]
    }
  }
}
```


Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Ejecute la acción [StopApplication](#) con la siguiente solicitud para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso CloudWatch de Logs con su aplicación, consulte [the section called "Configuración del registro"](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
```

```
"EnvironmentPropertyUpdates": {
  "PropertyGroups": [
    {
      "PropertyGroupId": "ProducerConfigProperties",
      "PropertyMap" : {
        "flink.stream.initpos" : "LATEST",
        "aws.region" : "us-west-2",
        "AggregationEnabled" : "false"
      }
    },
    {
      "PropertyGroupId": "ConsumerConfigProperties",
      "PropertyMap" : {
        "aws.region" : "us-west-2"
      }
    }
  ]
}
}
```

2. Ejecute la acción [UpdateApplication](#) con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción [UpdateApplication](#) de la AWS CLI.

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto

de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [the section called "Crear dos Amazon Kinesis Data Streams"](#).

```
{
  "ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

Paso siguiente

[Paso 4: Eliminar recursos de AWS](#)

Paso 4: Eliminar recursos de AWS

En esta sección se incluyen los procedimientos para eliminar los recursos de AWS creados en el tutorial de Introducción.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)

- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)
- [Paso siguiente](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.

6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Paso siguiente

[Paso 5: Sigüientes pasos](#)


Paso 5: Sigüientes pasos

Ahora que ha creado y ejecutado una aplicación básica de Managed Service para Apache Flink, consulte los siguientes recursos para conocer soluciones más avanzadas de Managed Service para Apache Flink.

- [Solución de datos de streaming de AWS para Amazon Kinesis](#): La solución de datos de streaming de AWS para Amazon Kinesis configura automáticamente los servicios de AWS necesarios para capturar, almacenar, procesar y entregar fácilmente datos de streaming. La solución ofrece varias opciones para resolver casos de uso de datos de streaming. La opción Managed Service for Apache Flink proporciona un ejemplo de ETL en end-to-end streaming que muestra una aplicación real que ejecuta operaciones analíticas con datos simulados de taxis de Nueva York. La solución configura todos los AWS recursos necesarios, como las funciones y políticas de IAM, un CloudWatch panel de control y alarmas. CloudWatch
- [Solución de datos de streaming de AWS para Amazon MSK](#): La solución de datos de streaming de AWS para Amazon MSK proporciona plantillas de AWS CloudFormation en las que los datos fluyen a través de los productores, el almacenamiento de streaming, los consumidores y los destinos.
- [Clickstream Lab con Apache Flink y Apache Kafka](#): Un laboratorio integral para casos de uso de secuencias de clics que utiliza Amazon Managed Streaming para Apache Kafka para

almacenamiento de streaming y aplicaciones de Managed Service para Apache Flink para procesamiento de flujos.

- [Taller de Amazon Managed Service para Apache Flink](#): en este taller, crearás una arquitectura de end-to-end streaming para ingerir, analizar y visualizar datos de streaming prácticamente en tiempo real. Usted se propuso mejorar las operaciones de una empresa de taxis de la ciudad de Nueva York. Usted analiza los datos de telemetría de una flota de taxis de la ciudad de Nueva York prácticamente en tiempo real para optimizar las operaciones de su flota.
- [Managed Service para Apache Flink: ejemplos](#): En esta sección de esta Guía para desarrolladores se proporcionan ejemplos de cómo crear y trabajar con aplicaciones en Managed Service para Apache Flink. Incluyen ejemplos de código e step-by-step instrucciones que le ayudarán a crear un servicio gestionado para las aplicaciones de Apache Flink y a comprobar sus resultados.
- [Aprenda Flink: formación práctica](#): formación oficial introductoria sobre Apache Flink que le permitirá empezar a crear aplicaciones escalables de ETL de streaming, análisis y basadas en eventos.

 Note

Tenga en cuenta que Managed Service para Apache Flink no es compatible con la versión de Apache Flink (1.12) utilizada en esta formación. Puede usar Flink 1.15.2 en Flink Managed Service para Apache Flink.

Introducción a Amazon Managed Service para Apache Flink (API de tabla)

Esta sección presenta una introducción a los conceptos fundamentales de Managed Service para Apache Flink y la API de tablas. Describe las opciones disponibles para crear y probar sus aplicaciones. También proporciona instrucciones para instalar las herramientas necesarias para completar los tutoriales de esta guía y crear su primera aplicación.

Temas

- [Componentes de la aplicación de Managed Service para Apache Flink](#)
- [Requisitos previos](#)
- [Crear y ejecutar una aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)
- [Siguiendo pasos](#)

Componentes de la aplicación de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Java/Apache Maven o Scala que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

La aplicación de Managed Service para Apache Flink tiene los siguientes componentes:

- **Propiedades de tiempo de ejecución:** puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- **Fuente de tabla:** la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un tema de Amazon MSK o similar. Para obtener más información, consulte [Fuentes de la API de tabla](#).
- **Funciones:** la aplicación procesa los datos mediante una o más funciones. Una función puede transformar, enriquecer o agregar datos.
- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data

Firehose, un tema de Amazon MSK, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores de la API de tabla](#).

Después de crear, compilar y empaquetar el código de la aplicación, debe cargar el paquete del código a un bucket de Amazon S3. Luego debe crear la aplicación de Managed Service para Apache Flink. Introduce la ubicación del paquete del código, un tema de Amazon MSK como origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos

Antes de iniciar este tutorial, complete los dos primeros pasos de [Introducción a Amazon Managed Service para Apache Flink \(DataStream API\)](#):

- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)

Para empezar, consulte [Crear una aplicación](#).

Crear y ejecutar una aplicación de Managed Service para Apache Flink

En este ejercicio, usted deberá crear una aplicación de Managed Service para Apache Flink con un tema de Amazon MSK como origen y un bucket de Amazon S3 como receptor.

Esta sección contiene los siguientes pasos.

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Paso siguiente](#)

Crear recursos dependientes

Antes de crear un Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Una nube privada virtual (VPC) basada en Amazon VPC y un clúster de Amazon MSK
- Un bucket de Amazon S3 para almacenar el código y los resultados de la aplicación (`ka-app-code-<username>`)

Crear una VPC y un clúster de Amazon MSK

Para crear una VPC y un clúster de Amazon MSK al que acceder desde la aplicación de Managed Service para Apache Flink, siga el tutorial [Introducción al uso de Amazon MSK](#).

Cuando complete el tutorial, tenga en cuenta lo siguiente:

- Registre la lista de servidores de arranque de su clúster. Puede obtener la lista de servidores de arranque con el siguiente comando, reemplazando `ClusterArn` con el nombre de recurso de Amazon (ARN) de su clúster de MSK:

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
  "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Al seguir los pasos de los tutoriales, asegúrese de usar la región de AWS seleccionada en el código, los comandos y las entradas de la consola.

Crear un bucket de Amazon S3

Puede crear el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear este recurso, consulte los siguientes temas:

- [¿Cómo se puede crear un bucket de S3?](#) en la Guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo `ka-app-code-<username>`.

Otros recursos

Al crear la aplicación, Managed Service for Apache Flink crea los siguientes CloudWatch recursos de Amazon si aún no existen:

- Un grupo de registro llamado `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flujo de registro llamado `kinesis-analytics-log-stream`.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el tema de Amazon MSK para que la aplicación los procese.

1. Conéctese a la instancia de cliente que creó en el [paso 4: Crear una máquina cliente](#) del tutorial [Introducción al uso de Amazon MSK](#).
2. Instale Python3, Pip y la biblioteca de Python de Kafka:

```
$ sudo yum install python37
$ curl -O https://bootstrap.pypa.io/get-pip.py
$ python3 get-pip.py --user
$ pip install kafka-python
```

3. Cree un archivo denominado `stock.py` con el siguiente contenido. Sustituya el valor `BROKERS` por la lista de agentes de arranque que registró anteriormente.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}
```



```
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

4. Más adelante en el tutorial, se ejecuta el script `stock.py` para enviar datos a la aplicación.

```
$ python3 stock.py
```

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java para este ejemplo está disponible en [GitHub](#)

Para descargar el código de la aplicación de Java

1. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/GettingStartedTable`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo [Project Object Model \(pom.xml\)](#) contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `StreamingJob.java` contiene el método `main` que define la funcionalidad de la aplicación.
- La aplicación utiliza un `FlinkKafkaConsumer` para leer el tema de Amazon MSK. El siguiente fragmento crea un objeto de `FlinkKafkaConsumer`:

```
final FlinkKafkaConsumer<StockRecord> consumer = new
    FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
    kafkaProps);
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos usando objetos `StreamExecutionEnvironment` y `TableEnvironment`.
- La aplicación crea conectores de origen y recepción mediante propiedades dinámicas de la aplicación, de modo que usted puede especificar los parámetros de la aplicación (como el bucket de S3) sin tener que volver a compilar el código.

```
//read the parameters from the Managed Service for Apache Flink environment
Map<String, Properties> applicationProperties =
    KinesisAnalyticsRuntime.getApplicationProperties();
Properties flinkProperties = null;

String kafkaTopic = parameter.get("kafka-topic", "AWSKafkaTutorialTopic");
String brokers = parameter.get("brokers", "");
String s3Path = parameter.get("s3Path", "");

if (applicationProperties != null) {
    flinkProperties = applicationProperties.get("FlinkApplicationProperties");
}

if (flinkProperties != null) {
    kafkaTopic = flinkProperties.get("kafka-topic").toString();
    brokers = flinkProperties.get("brokers").toString();
    s3Path = flinkProperties.get("s3Path").toString();
}
```

Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Note

Al crear la aplicación, le recomendamos encarecidamente que cree y ejecute la aplicación Managed Service para Apache Flink en la misma región que el clúster de Amazon MSK. Esto se debe a que el conector Flink Kafka está optimizado de forma predeterminada para entornos de baja latencia. Si necesita consumir desde un clúster de Kafka entre regiones,

considere la posibilidad de aumentar el valor de configuración de `receive.buffer.byte`, por ejemplo, 2097152.

Para obtener más información, consulte [Configuraciones personalizadas de MSK](#).

Compilar el código de la aplicación

En esta sección, se utiliza el compilador Apache Maven para crear el código de Java para la aplicación. Para obtener más información sobre la instalación de Apache Maven y el Java Development Kit (JDK), consulte [Requisitos previos para completar los ejercicios](#).

Para compilar el código de la aplicación

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar el código de una de las dos formas siguientes:
 - Utilice la herramienta de línea de comandos de Maven. Cree su archivo JAR ejecutando el siguiente comando en el directorio que contiene el archivo `pom.xml`:

```
mvn package -Dflink.version=1.15.3
```

- Use el entorno de desarrollo. Consulte la documentación de su entorno de desarrollo para obtener más información.

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Puede cargar el paquete como un archivo JAR o puede comprimir el paquete y cargarlo como un archivo ZIP. Si crea su aplicación mediante la AWS CLI, debe especificar el tipo de contenido del código (JAR o ZIP).

2. Si hay errores al compilar, verifique que la variable de entorno `JAVA_HOME` se ha configurado correctamente.

Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Cargar el código de Java de streaming de Apache Flink

En esta sección, creará un bucket de Amazon S3 y cargará el código de la aplicación.

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket.
3. Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. En la consola de Amazon S3, elija el *<username>*bucket ka-app-code- y elija Upload.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior. Elija Siguiente.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.

- Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para añadir los permisos para acceder al bucket de Amazon S3.

Para editar la política de IAM para añadir los permisos para el bucket de S3

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
      "Effect": "Allow",
      "Action": [
        "s3:Abort*",
        "s3:DeleteObject*",
```

```

        "s3:GetObject*",
        "s3:GetBucket*",
        "s3:List*",
        "s3:ListBucket",
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:s3:::ka-app-code-<username>",
        "arn:aws:s3:::ka-app-code-<username>/*"
    ]
},
{
    "Sid": "DescribeLogGroups",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
},
{
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogStreams"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
},
{
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
}
]

```

}

Configurar la aplicación

Utilice el siguiente procedimiento para configurar la aplicación.

Para configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, elija Crear grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
FlinkApplicationProperties	kafka-topic	AWSKafkaTutorialTopic
FlinkApplicationProperties	brokers	<i>Your Amazon MSK cluster's Bootstrap Brokers list</i>
FlinkApplicationProperties	s3Path	ka-app-code- <i><username></i>
FlinkApplicationProperties	security.protocol	SSL
FlinkApplicationProperties	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto

ID de grupo	Clave	Valor
		/lib/security/cacerts
FlinkApplicationProperties	ssl.truststore.password	changeit

- En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
- Para el CloudWatch registro, active la casilla Activar.
- En la sección Nube privada virtual (VPC), elija Configuración de VPC basada en el clúster de Amazon MSK. Elija AWSKafkaTutorialCluster.
- Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Ejecutar la aplicación

Utilice el siguiente procedimiento para ejecutar la aplicación.

Para ejecutar la aplicación

- En la MyApplication página, selecciona Ejecutar. Confirme la acción.
- Cuando la aplicación se está ejecutando, actualice la página. La consola muestra el Gráfico de la aplicación.
- Desde su cliente Amazon EC2, ejecute el script de Python que creó anteriormente para escribir registros en el clúster de Amazon MSK para que la aplicación los procese:


```
$ python3 stock.py
```

Detener la aplicación

Para detener la aplicación, en la MyApplication página, seleccione Detener. Confirme la acción.

Paso siguiente

[Eliminación de recursos de AWS](#)

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Introducción (API de tablas).

Este tema contiene las siguientes secciones.

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Elimine su clúster de Amazon MSK](#)
- [Eliminar su VPC](#)
- [Eliminar objetos y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)
- [Paso siguiente](#)

Elimine su aplicación de Managed Service para Apache Flink

Utilice el siguiente procedimiento para eliminar la aplicación.

Para eliminar la aplicación

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Elimine su clúster de Amazon MSK

Para eliminar su clúster de Amazon MSK, siga el [paso 8: Eliminar el clúster de Amazon MSK](#) de la [Guía para desarrolladores de Amazon Managed Streaming para Apache Kafka](#).

Eliminar su VPC

Para eliminar su Amazon VPC, haga lo siguiente:

- Abra la Consola de Amazon VPC.
- Elija su VPC.
- En Actions (Acciones), elija Delete VPC (Eliminar VPC).

Eliminar objetos y el bucket de Amazon S3

Siga el siguiente procedimiento para eliminar los objetos y el bucket de S3.

Para eliminar los objetos y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el compartimento ka-app-code-<username>.
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

Utilice el siguiente procedimiento para eliminar sus recursos de IAM.

Para eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication

8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

Utilice el siguiente procedimiento para eliminar CloudWatch los recursos.

Para eliminar sus CloudWatch recursos

1. Abre la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Paso siguiente

[Sigüientes pasos](#)

Sigüientes pasos

Ahora que ha creado y ejecutado una aplicación de Managed Service para Apache Flink que utiliza la API de tablas, consulte [Paso 5: Sigüientes pasos](#) en [Introducción a Amazon Managed Service para Apache Flink \(DataStream API\)](#).

Introducción a Amazon Managed Service para Apache Flink para Python

Esta sección presenta una introducción a los conceptos fundamentales de un Managed Service para Apache Flink utilizando Python y la API de tabla. Describe las opciones disponibles para crear y probar sus aplicaciones. También proporciona instrucciones para instalar las herramientas necesarias para completar los tutoriales de esta guía y crear su primera aplicación.

Temas

- [Introducción a Pyflink: el intérprete de Python para Apache | Amazon Web Services](#)
- [Componentes de la aplicación de Managed Service para Apache Flink](#)
- [Requisitos previos](#)
- [Crear y ejecutar una aplicación de Managed Service para Apache Flink para Python](#)
- [Eliminación de recursos de AWS](#)

Note

Si está desarrollando la aplicación Python Flink en una Mac nueva con el chip Apple Silicon, es posible que encuentre algunos [problemas conocidos](#) con las dependencias de Python de PyFlink la versión 1.15. En este caso, recomendamos ejecutar el intérprete de Python en Docker. Para obtener step-by-step instrucciones, consulta el [desarrollo de la PyFlink versión 1.15 en Apple Silicon Mac](#).

Introducción a Pyflink: el intérprete de Python para Apache | Amazon Web Services

Antes de comenzar, le recomendamos que vea el siguiente vídeo:

[Introducción a Pyflink: el intérprete de Python para Apache | Amazon Web Services](#)

Componentes de la aplicación de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Python que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

La aplicación de Managed Service para Apache Flink tiene los siguientes componentes:

- **Propiedades de tiempo de ejecución:** puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- **Fuente de tabla:** la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un tema de Amazon MSK o similar. Para obtener más información, consulte [Fuentes de la API de tabla](#).
- **Funciones:** la aplicación procesa los datos mediante una o más funciones. Una función puede transformar, enriquecer o agregar datos.
- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose, un tema de Amazon MSK, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores de la API de tabla](#).

Después de crear y empaquetar el código de la aplicación, debe cargar el paquete del código a un bucket de Amazon S3. Luego debe crear la aplicación de Managed Service para Apache Flink. Introduce la ubicación del paquete del código, el origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos

Antes de iniciar este tutorial, complete los dos primeros pasos de [Introducción a Amazon Managed Service para Apache Flink \(DataStream API\)](#):

- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)

Para empezar, consulte [Crear una aplicación](#).

Crear y ejecutar una aplicación de Managed Service para Apache Flink para Python

En este ejercicio, se creará una aplicación de Managed Service para Apache Flink para Python con una secuencia de Kinesis como origen y receptor.

Esta sección contiene los siguientes pasos.

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Crear y examinar el código de Python de streaming de Apache Flink](#)
- [Añadir dependencias de terceros a las aplicaciones de Python](#)
- [Cargar el código de Python de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Paso siguiente](#)

Crear recursos dependientes

Antes de crear un Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de Kinesis para entrada y salida.
- Un bucket de Amazon S3 para almacenar el código y los resultados de la aplicación (ka-app-code-*<username>*)

Crear dos flujos de Kinesis

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, cree dos flujos de datos de Kinesis (ExampleInputStream y ExampleOutputStream). Su aplicación utiliza estos flujos para los flujos de origen y destino de la aplicación.

Puede crear estos flujos mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para crear flujos de datos (AWS CLI)

1. Para crear el primer flujo (ExampleInputStream), utilice el siguiente comando de la AWS CLI `create-stream` de Amazon Kinesis.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

2. Para crear el segundo flujo que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a ExampleOutputStream.

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

Crear un bucket de Amazon S3

Puede crear el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear este recurso, consulte los siguientes temas:

- [¿Cómo se puede crear un bucket de S3?](#) en la Guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Otros recursos

Al crear la aplicación, Managed Service for Apache Flink crea los siguientes CloudWatch recursos de Amazon si aún no existen:

- Un grupo de registro llamado `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flujo de registro llamado `kinesis-analytics-log-stream`.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

Note

El script de Python en esta sección usa AWS CLI. Debe configurar sus AWS CLI para usar las credenciales de su cuenta y la región predeterminada. Para configurar su AWS CLI, escriba lo siguiente:

```
aws configure
```

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
```



```
print(data)
kinesis_client.put_record(
    StreamName=stream_name,
    Data=json.dumps(data),
    PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Crear y examinar el código de Python de streaming de Apache Flink

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/python/GettingStarted`.

El código de la aplicación se encuentra en el archivo `getting_started.py`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de tabla de Kinesis para leer del flujo de origen. El siguiente fragmento llama a la función `create_table` para crear el origen de la tabla de Kinesis:

```
table_env.execute_sql(
    create_table(output_table_name, output_stream, output_region)
```

La función `create_table` utiliza un comando SQL para crear una tabla respaldada por el origen de streaming:

```
def create_table(table_name, stream_name, region, stream_initpos = None):
    init_pos = "\n'scan.stream.initpos' = '{0}',".format(stream_initpos) if
    stream_initpos is not None else ''

    return """ CREATE TABLE {0} (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time TIMESTAMP(3),
        WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{1}',
        'aws.region' = '{2}',{3}
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """.format(table_name, stream_name, region, init_pos)
}
```

- La aplicación crea dos tablas y, a continuación, escribe el contenido de una tabla en la otra.

```
# 2. Creates a source table from a Kinesis Data Stream
table_env.execute_sql(
    create_table(input_table_name, input_stream, input_region)
)

# 3. Creates a sink table writing to a Kinesis Data Stream
table_env.execute_sql(
    create_table(output_table_name, output_stream, output_region, stream_initpos)
)

# 4. Inserts the source table data into the sink table
table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
    .format(output_table_name, input_table_name))
```

- La aplicación utiliza el conector Flink, del archivo [flink-sql-connector-kinesis_2.12/1.15.2](#).

Añadir dependencias de terceros a las aplicaciones de Python

Cuando se utilicen paquetes de Python de terceros (como [boto3](#)), tendrá que añadir sus dependencias transitivas y las propiedades necesarias para adaptarse a estas dependencias. En un nivel superior, para PyPi las dependencias, puede copiar los archivos y carpetas que se encuentran dentro de la `site-packages` carpeta de entornos de Python para crear una estructura de directorios como la siguiente:

```
PythonPackages
#  README.md
#  python-packages.py
#
####my_deps
    ####boto3
    #  #  session.py
    #  #  utils.py
    #  #  ...
    #
    ####botocore
    #  #  args.py
    #  #  auth.py
    #  #  ...
    ####mynonpypimodule
    #  #  mymodulefile1.py
    #  #  mymodulefile2.py
    #  #  ...
####lib
#  #  flink-sql-connector-kinesis-1.15.2.jar
#  #  ...
...
```

Para añadir boto3 como una dependencia de terceros:

1. Cree un entorno de Python independiente (conda o similar) en su máquina local con las dependencias necesarias.
2. Anote la lista inicial de paquetes de la carpeta `site_packages` de ese entorno.
3. `pip-install` todas las dependencias necesarias para su aplicación.
4. Anote los paquetes que se agregaron a la carpeta `site_packages` después del paso 3 indicado anteriormente. Estas son las carpetas que debe incluir en su paquete (debajo de la carpeta `my_deps`), organizadas como se muestra arriba. Esto le permitirá capturar una

diferencia de los paquetes entre los pasos 2 y 3 para identificar las dependencias de paquetes adecuadas para su aplicación.

- Indique `my_deps/` como argumento para la propiedad `pyFiles` en el grupo de propiedades `kinesis.analytics.flink.run.options`, tal como se describe a continuación para la propiedad `jarfiles`. Flink también le permite especificar las dependencias de Python mediante la función [add_python_file](#), pero es importante tener en cuenta que solo necesita especificar una u otra, no ambas.

Note

No es necesario que le ponga un nombre a la carpeta `my_deps`. Lo importante es registrar las dependencias utilizando `pyFiles` o, bien, `add_python_file`. Puede encontrar un ejemplo en [Cómo usar boto3 en pyFlink](#).

Cargar el código de Python de streaming de Apache Flink


En esta sección, creará un bucket de Amazon S3 y cargará el código de la aplicación.

Para cargar el código de la aplicación mediante la consola:

- Utilice la aplicación de compresión que prefiera para `getting-started.py` comprimir los archivos https://mvnrepository.com/artifact/org.apache.flink/flink-sql-connector-kinesis_2.12/1.15.2. Dé el nombre `myapp.zip` al archivo. Si incluye la carpeta externa en su archivo, debe incluirla en la ruta con el código de su(s) archivo(s) de configuración: `GettingStarted/getting-started.py`.
- Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
- Elija Crear bucket.
- Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
- En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
- En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
- Elija Crear bucket.
- En la consola de Amazon S3, elija el `<username>bucket ka-app-code-` y elija Upload.


9. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `myapp.zip` que creó en el paso anterior. Elija Siguiente.
10. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

Para cargar el código de la aplicación mediante AWS CLI:

 Note

No utilice las características de compresión de Finder (MacOS) o del Explorador de Windows (Windows) para crear el archivo `myapp.zip`. Esto puede dar lugar a que el código de la aplicación no sea válido.

1. Utilice la aplicación de compresión que prefiera para `streaming-file-sink.py` comprimir los archivos <https://mvnrepository.com/artifact/org.apache.flink/flink-sql-connector-kinesis-2.12/1.15.2>.

 Note

No utilice las características de compresión de Finder (MacOS) o del Explorador de Windows (Windows) para crear el archivo `myapp.zip`. Esto puede dar lugar a que el código de la aplicación no sea válido.

2. [Utilice la aplicación de compresión que prefiera para comprimir los archivos https://mvnrepository.com/artifact/org.apache.flink/1.15.2.getting-started.py](https://mvnrepository.com/artifact/org.apache.flink/1.15.2/getting-started.py) `flink-sql-connector-kinesis` Dé el nombre `myapp.zip` al archivo. Si incluye la carpeta externa en su archivo, debe incluirla en la ruta con el código de su(s) archivo(s) de configuración: `GettingStarted/getting-started.py`.
3. Ejecute el siguiente comando:

```
$ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
```

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

Utilice el siguiente procedimiento para configurar la aplicación.

Para configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **myapp.zip**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, elija Añadir grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

Seleccione Guardar.

6. En Propiedades, elija Añadir grupo nuevamente.
7. Introduzca lo siguiente:


ID de grupo	Clave	Valor
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. En Propiedades, elija Añadir grupo nuevamente. En Nombre de grupo, introduzca **kinesis.analytics.flink.run.options**. Este grupo de propiedades especiales le indica a su aplicación dónde encontrar sus recursos de código. Para obtener más información, consulte [Especificar sus archivos de código](#).

9. Introduzca lo siguiente:

ID de grupo	Clave	Valor
kinesis.analytics.flink.run.options	python	getting-started.py
kinesis.analytics.flink.run.options	jarfile	flink-sql-connector-kinesis-1.15.2.jar

10. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
11. Para el CloudWatch registro, active la casilla Activar.
12. Seleccione Actualizar.

 Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: /aws/kinesis-analytics/MyApplication
- Flujo de registro: kinesis-analytics-log-stream

Editar la política de IAM

Edite la política de IAM para añadir los permisos para acceder al bucket de Amazon S3.

Para editar la política de IAM para añadir los permisos para el bucket de S3

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.

4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (*012345678901*) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/myapp.zip"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ]
    }
  ]
}
```

```

    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

Para detener la aplicación, en la MyApplication página, seleccione Detener. Confirme la acción.

Paso siguiente

[Eliminación de recursos de AWS](#)

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Introducción (Python).

Este tema contiene las siguientes secciones.

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar objetos y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

Utilice el siguiente procedimiento para eliminar la aplicación.

Para eliminar la aplicación

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar objetos y el bucket de Amazon S3

Siga el siguiente procedimiento para eliminar los objetos y el bucket de S3.

Para eliminar los objetos y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el <username>cubo ka-app-code-.
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

Utilice el siguiente procedimiento para eliminar sus recursos de IAM.

Para eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

Utilice el siguiente procedimiento para eliminar CloudWatch los recursos.

Para eliminar sus CloudWatch recursos

1. Abre la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Introducción (Scala)

Note

A partir de la versión 1.15, Flink es gratuito para Scala. Las aplicaciones ahora pueden usar la API de Java desde cualquier versión de Scala. Flink sigue utilizando Scala internamente en algunos componentes clave, pero no lo expone al cargador de clases del código de usuario. Por eso, los usuarios deben agregar las dependencias de Scala a sus archivos jar. Para obtener más información sobre los cambios de Scala en Flink 1.15, consulte [Scala Free in One Fifteen](#).

En este ejercicio, se creará una aplicación de Managed Service para Apache Flink para Scala con un flujo de Kinesis como origen y recepción.

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en la secuencia de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Compilar y cargar el código de la aplicación](#)
- [Crear y ejecutar la aplicación \(consola\)](#)
- [Compilar y ejecutar la aplicación \(CLI\)](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de Kinesis para entrada y salida.
- Un bucket de Amazon S3 para almacenar el código de la aplicación (ka-app-code-*<username>*)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a sus flujos de datos **ExampleInputStream** y **ExampleOutputStream**.

Para crear secuencias de datos (AWS CLI)

- Para crear el primer flujo (ExampleInputStream), utilice el siguiente comando de AWS CLI para crear flujo de Amazon Kinesis.

```
aws kinesis create-stream \  
  --stream-name ExampleInputStream \  
  --shard-count 1 \  
  --region us-west-2 \  
  --profile adminuser
```

- Para crear la segunda secuencia que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a ExampleOutputStream.

```
aws kinesis create-stream \  
  --stream-name ExampleOutputStream \  
  --shard-count 1 \  
  --region us-west-2 \  
  --profile adminuser
```

- [¿Cómo se puede crear un bucket de S3?](#) en la Guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Otros recursos

Al crear la aplicación, Managed Service para Apache Flink crea los siguientes recursos de Amazon CloudWatch si aún no existen:

- Un grupo de registro llamado `/AWS/KinesisAnalytics-java/MyApplication`
- Un flujo de registro llamado `kinesis-analytics-log-stream`

Escribir registros de muestra en la secuencia de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en la secuencia para que la aplicación realice el procesamiento.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

Note

El script de Python en esta sección usa AWS CLI. Debe configurar sus AWS CLI para usar las credenciales de su cuenta y la región predeterminada. Para configurar su AWS CLI, escriba lo siguiente:

```
aws configure
```

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
```

```
kinesis_client.put_record(  
    StreamName=stream_name,  
    Data=json.dumps(data),  
    PartitionKey="partitionkey")  
  
if __name__ == '__main__':  
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación de Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo `build.sbt` contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.scala` contiene el método principal que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer del flujo de origen. El siguiente fragmento crea el origen de Kinesis:


```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
    defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

La aplicación también utiliza un receptor de Kinesis para escribir en el flujo de resultado. El siguiente fragmento crea el receptor de Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
    defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos mediante un objeto `StreamExecutionEnvironment`.
- La aplicación crea conectores de origen y recepción mediante las propiedades dinámicas de la aplicación. Las propiedades de tiempo de ejecución de la aplicación se leen para configurar los conectores. Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar y cargar el código de la aplicación

En esta sección, compilará y cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

Compilar el código de la aplicación

En esta sección, utilizará la herramienta de creación [SBT](#) para crear el código Scala para la aplicación. Para instalar SBT, consulte [Instalar sbt con configuración de cs](#). También deberá instalar el Kit de desarrollo de Java (JDK). Consulte los [Requisitos previos para completar los ejercicios](#).

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar su código con SBT:

```
sbt assembly
```

2. Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

Cargar el código de Scala de streaming de Apache Flink

En esta sección, creará un bucket de Amazon S3 y cargará el código de la aplicación.

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3>.
2. Elija Crear bucket
3. Escriba `ka-app-code- \langle username \rangle` en el campo Bucket name (Nombre del bucket). Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Next (Siguiente).
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. Abra el bucket `ka-app-code- \langle username \rangle` y elija Cargar.
8. En el paso Select files (Seleccionar archivos), elija Add files (Añadir archivos). Vaya al archivo `getting-started-scala-1.0.jar` que creó en el paso anterior.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación de

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Application name (Nombre de la application), escriba **MyApplication**.
 - En Descripción, escriba **My scala test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).
4. Para Permisos de acceso, seleccione Crear o actualizar rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
5. Elija Create application.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

Utilice el siguiente procedimiento para configurar la aplicación.

Para configurar la aplicación

1. En la página MyApplication (Mi aplicación), elija Configure (Configurar).
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - Para Ruta al objeto de Amazon S3, introduzca **getting-started-scala-1.0.jar..**
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, elija Añadir grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

Seleccione Save.

6. En Propiedades, elija Añadir grupo nuevamente.
7. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

8. En Monitoring (Monitorización), asegúrese de que el Monitoring metrics level (Nivel de métricas de monitorización) se ha establecido en Application (Aplicación).
9. En Registro de CloudWatch, elija la casilla de verificación Habilitar.
10. Elija Actualizar.

Note

Al activar el registro de Amazon CloudWatch, Managed Service para Apache Flink crea un grupo de registro y un flujo de registro. Los nombres de estos recursos son los siguientes:

- Secuencia de registros: /aws/kinesis-analytics/MyApplication
- Secuencia de registros: kinesis-analytics-log-stream

Editar la política de IAM

Edite la política de IAM para añadir los permisos para acceder al bucket de Amazon S3.

Para editar la política de IAM para añadir los permisos para el bucket de S3

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola ha creado en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (**012345678901**) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
```

```

        "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
    ]
},
{
    "Sid": "DescribeLogGroups",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
},
{
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogStreams"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
},
{
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
},
{
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
},
{
    "Sid": "WriteOutputStream",

```

```
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
  ]
}
```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

Para detener la aplicación, en la página MyApplication, seleccione Detener. Confirme la acción.

Compilar y ejecutar la aplicación (CLI)

En esta sección se utiliza la AWS Command Line Interface para crear y ejecutar la aplicación de Managed Service para Apache Flink. Utilice el comando AWS CLI `kinesisanalyticsv2` para crear aplicaciones de Managed Service para Apache Flink e interactuar con ellas.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción de lectura en el flujo de origen y otra que concede permisos para las acciones de escritura en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

AKReadSourceStreamWriteSinkStream. Reemplace **username** por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Reemplace el ID de la cuenta en el nombre de recurso de Amazon (ARN) (**012345678901**) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutLogEvents",
```



```

    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Para obtener instrucciones paso a paso para crear una política de permisos, consulte [Tutorial: Crear y asociar su primera política administrada por el cliente](#) en la Guía del usuario de IAM.

Crear una política de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Asociará la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, seleccione Roles y, a continuación, seleccione Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS
4. En Choose the service that will use this role (Elegir el servicio que usará este rol), elija Kinesis.
5. En Seleccione su caso de uso, elija Managed Service para Apache Flink.
6. Elija Siguiente: Permisos.
7. En la página Attach permissions policies, elija Next: Review. Asociará políticas de permisos después de crear el rol.
8. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado MF-stream-rw-role. A continuación, actualice las políticas de confianza y permisos para el rol

9. Asocie una política de permisos al rol.

Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Para asociar la política que ha creado en el paso anterior: [Crear una política de permisos](#).

- a. En la página Summary (Resumen), elija la pestaña Permissions (Permisos).
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la política AKReadSourceStreamWriteSinkStream y, a continuación, elija Asociar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para obtener instrucciones paso a paso para crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía de usuario de IAM.

Crear la aplicación

Copie el siguiente código JSON a un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (nombre de usuario) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de muestra (012345678901) en el rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "getting_started",
  "ApplicationDescription": "Scala getting started application",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "getting-started-scala-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleInputStream",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleOutputStream"
          }
        }
      ]
    }
  }
}
```

```
    }
  },
  "CloudWatchLoggingOptions": [
    {
      "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
    }
  ]
}
```

Ejecute [CreateApplication](#) con la siguiente solicitud para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, utilizará la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Copie el siguiente código JSON a un archivo denominado `start_request.json`.

```
{
  "ApplicationName": "getting_started",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

2. Ejecute la acción `StartApplication` con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

La aplicación se está ejecutando. Puede comprobar las métricas de Managed Service para Apache Flink en la consola de Amazon CloudWatch para verificar que la aplicación funciona.

Detener la aplicación

En esta sección, utilizará la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Copie el siguiente código JSON a un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "s3_sink"
}
```

2. Ejecute la acción `StopApplication` con la solicitud anterior para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se ha detenido.

Añadir una opción de registro de CloudWatch

Puede utilizar la AWS CLI para añadir un flujo de registro de Amazon CloudWatch a su aplicación.

Para obtener información sobre el uso de CloudWatch Logs con su aplicación, consulte

[Configuración del registro de aplicaciones](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Copie el siguiente código JSON a un archivo denominado `update_properties_request.json`.

```
{
  "ApplicationName": "getting_started",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
```

```
"PropertyGroups": [  
  {  
    "PropertyGroupId": "ConsumerConfigProperties",  
    "PropertyMap" : {  
      "aws.region" : "us-west-2",  
      "stream.name" : "ExampleInputStream",  
      "flink.stream.initpos" : "LATEST"  
    }  
  },  
  {  
    "PropertyGroupId": "ProducerConfigProperties",  
    "PropertyMap" : {  
      "aws.region" : "us-west-2",  
      "stream.name" : "ExampleOutputStream"  
    }  
  }  
]  
}
```

2. Ejecute la acción `UpdateApplication` con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://  
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción de CLI [UpdateApplication](#).

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto

de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [Crear recursos dependientes](#).

```
{
  "ApplicationName": "getting_started",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-<username>",
          "FileKeyUpdate": "getting-started-scala-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana de salto de tamaño constante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [Eliminar sus recursos de CloudWatch](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel de Kinesis Data Streams, elija ExampleInputStream.
3. En la página ExampleInputStream, elija Eliminar flujo de Kinesis y, a continuación, confirme la eliminación.
4. En la página Flujos de Kinesis, elija ExampleOutputStream, elija Acciones, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3>.
2. Elija el bucket ka-app-code-**<username>**.
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elija la política kinesis-analytics-service-MyApplication-us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija el rol kinesis-analytics-MyApplication-us-west-2.
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

Eliminar sus recursos de CloudWatch

1. Abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registro /aws/kinesis-analytics/MyApplication.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Creación de aplicaciones de Managed Service para Apache Flink con Apache Beam

Puede usar el marco [Apache Beam](#) con su aplicación Managed Service para Apache Flink para procesar datos de streaming. Las aplicaciones de Managed Service para Apache Flink que utilizan Apache Beam, utilizan el [ejecutor Apache Flink](#) para ejecutar las canalizaciones de Beam.

Para obtener un tutorial sobre cómo usar Apache Beam en una aplicación de Managed Service para Apache Flink, consulte [Uso de CloudFormation con Managed Service para Apache Flink](#).

Este tema contiene las siguientes secciones:

- [Uso de Apache Beam con Managed Service para Apache Flink](#)
- [Capacidades de Beam](#)
- [Crear una aplicación con Apache Beam](#)

Uso de Apache Beam con Managed Service para Apache Flink

Note lo siguiente sobre el uso del ejecutor de Apache Flink con Managed Service para Apache Flink:

- Las métricas de Apache Beam no se pueden ver en la consola de Managed Service para Apache Flink.
- Apache Beam solo es compatible con el servicio gestionado para las aplicaciones de Apache Flink que utilizan la versión 1.8 o superior de Apache Flink. Apache Beam no es compatible con las aplicaciones de Managed Service para Apache Flink que utilizan la versión 1.6 de Apache Flink.

Capacidades de Beam

Managed Service para Apache Flink es compatible con las mismas capacidades de Apache Beam que el ejecutor Apache Flink. Para obtener información sobre las características compatibles con el ejecutor Apache Flink, consulte la [matriz de compatibilidad de Beam](#).

Le recomendamos que pruebe su aplicación Apache Flink en el servicio Managed Service para Apache Flink para comprobar que todas las características que su aplicación necesita sean compatibles.

Crear una aplicación con Apache Beam

En este ejercicio, creará una aplicación de Managed Service para Apache Flink que transforma datos usando [Apache Beam](#). Apache Beam es un modelo de programación para procesar datos de streaming. Para obtener más información sobre el uso de Apache Beam con Managed Service para Apache Flink, consulte [Uso de Apache Beam](#).

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en la secuencia de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)
- [Pasos siguientes](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de datos de Kinesis (ExampleInputStream y ExampleOutputStream)
- Un bucket de Amazon S3 para almacenar el código de la aplicación (ka-app-code-*<username>*)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a sus flujos de datos **ExampleInputStream** y **ExampleOutputStream**.
- [¿Cómo se puede crear un bucket de S3?](#) en la Guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en la secuencia de entrada

En esta sección, se utiliza un script de Python para escribir cadenas asignadas al azar al flujo para que la aplicación realice el procesamiento.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `ping.py` con el siguiente contenido:

```
import json
import boto3
import random

kinesis = boto3.client('kinesis')

while True:
    data = random.choice(['ping', 'telnet', 'ftp', 'tracert', 'netstat'])
    print(data)
    kinesis.put_record(
        StreamName="ExampleInputStream",
        Data=data,
        PartitionKey="partitionkey")
```

2. Ejecute el script `ping.py`:

```
$ python ping.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación de Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/Beam`.

El código de la aplicación se encuentra en el archivo `BasicBeamStreamingJob.java`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza el Apache Beam [ParDo](#) para procesar los registros entrantes al invocar una función de transformación personalizada llamada `PingPongFn`.

El código para invocar la función `PingPongFn` es el siguiente:

```
.apply("Pong transform",  
      ParDo.of(new PingPongFn()))
```

- Las aplicaciones de Managed Service para Apache Flink que usan Apache Beam requieren los siguientes componentes. Si no incluye estos componentes y versiones en su `pom.xml`, la aplicación carga las versiones incorrectas desde las dependencias del entorno y, dado que las versiones no coinciden, la aplicación se bloquea durante el tiempo de ejecución.

```
<jackson.version>2.10.2</jackson.version>  
...  
<dependency>  
  <groupId>com.fasterxml.jackson.module</groupId>  
  <artifactId>jackson-module-jaxb-annotations</artifactId>  
  <version>2.10.2</version>  
</dependency>
```

- La función de transformación `PingPongFn` pasa los datos de entrada al flujo de salida, a menos que los datos de entrada sean ping, en cuyo caso emite la cadena `pong\n` al flujo de salida.

El código de la función de transformación es el siguiente:

```
private static class PingPongFn extends DoFn<KinesisRecord, byte[]> {
    private static final Logger LOG = LoggerFactory.getLogger(PingPongFn.class);

    @ProcessElement
    public void processElement(ProcessContext c) {
        String content = new String(c.element().getDataAsBytes(),
            StandardCharsets.UTF_8);
        if (content.trim().equalsIgnoreCase("ping")) {
            LOG.info("Ponged!");
            c.output("pong\n".getBytes(StandardCharsets.UTF_8));
        } else {
            LOG.info("No action for: " + content);
            c.output(c.element().getDataAsBytes());
        }
    }
}
```

Compilar el código de la aplicación

Para compilar la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale Java y Maven. Para obtener más información, consulte [Requisitos previos](#) en el tutorial de [Cómo empezar \(DataStream API\)](#).
2. Compile la aplicación con el siguiente comando:

```
mvn package -Dflink.version=1.15.3 -Dflink.version.minor=1.8
```

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Al compilar la aplicación, se crea el archivo JAR de la aplicación (`target/basic-beam-app-1.0.jar`).

Cargar el código de Java de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. En la consola de Amazon S3, elija el bucket `ka-app-code-<username>` y seleccione Subir.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `basic-beam-app-1.0.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación de

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink.

Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para Permisos de acceso, seleccione Crear o actualizar rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesis-analytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola ha creado en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/basic-beam-app-1.0.jar"
      ]
    },
    {
```



```

        "Sid": "DescribeLogStreams",
        "Effect": "Allow",
        "Action": "logs:DescribeLogStreams",
        "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
    },
    {
        "Sid": "PutLogEvents",
        "Effect": "Allow",
        "Action": "logs:PutLogEvents",
        "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    },
    {
        "Sid": "ListCloudwatchLogGroups",
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogGroups"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:*"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
]
}

```

Configurar la aplicación

1. En la página MyApplication, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - Para Ruta al objeto de Amazon S3, introduzca **basic-beam-app-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Introduzca lo siguiente:

ID de grupo	Clave	Valor
BeamApplicationProperties	InputStreamName	ExampleInputStream
BeamApplicationProperties	OutputStreamName	ExampleOutputStream
BeamApplicationProperties	AwsRegion	us-west-2

5. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
6. En Registro de CloudWatch, seleccione la casilla de verificación Habilitar.
7. Elija Actualizar.

Note

Cuando se elige habilitar registros de CloudWatch, Managed Service para Apache Flink crea un grupo de registro y un flujo de registro para usted. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Puede comprobar las métricas de Managed Service para Apache Flink en la consola de CloudWatch para verificar que la aplicación funciona.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana de salto de tamaño constante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [Eliminar sus recursos de CloudWatch](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel de Kinesis Data Streams, elija ExampleInputStream.

3. En la página ExampleInputStream, elija Eliminar flujo de Kinesis y, a continuación, confirme la eliminación.
4. En la página Flujos de Kinesis, elija ExampleOutputStream, elija Acciones, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3>.
2. Elija el bucket ka-app-code-**<username>**.
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elija la política kinesis-analytics-service-MyApplication-us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija el rol kinesis-analytics-MyApplication-us-west-2.
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

Eliminar sus recursos de CloudWatch

1. Abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registro /aws/kinesis-analytics/MyApplication.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Pasos siguientes

Ya que ha creado y ejecutado una aplicación básica de Managed Service para Apache Flink que transforma los datos usando Apache Beam, consulte la siguiente aplicación para encontrar un ejemplo de una solución más avanzada de Managed Service para Apache Flink.

- [Taller sobre streaming de Managed Service para Apache Flink](#): en este taller, analizamos un ejemplo integral que combina aspectos de transmisión por lotes y streaming en una canalización uniforme de Apache Beam.

Talleres de formación, laboratorios e implementaciones de soluciones

Los siguientes end-to-end ejemplos muestran las soluciones avanzadas de Managed Service for Apache Flink.

Temas

- [Desarrollo de aplicaciones de Apache Flink localmente antes de implementarlas en Managed Service para Apache Flink](#)
- [Detección de eventos con Managed Service para Apache Flink Studio](#)
- [Solución de datos de streaming de AWS para Amazon Kinesis](#)
- [Clickstream Lab con Apache Flink y Apache Kafka](#)
- [Escalado personalizado utilizando el escalado automático de aplicaciones](#)
- [CloudWatch Panel de control de Amazon](#)
- [Solución de datos de streaming de AWS para Amazon MSK](#)
- [Más servicios gestionados para Apache Flink Solutions en GitHub](#)

Desarrollo de aplicaciones de Apache Flink localmente antes de implementarlas en Managed Service para Apache Flink

Este taller le mostrará los conceptos básicos para empezar a desarrollar aplicaciones de Apache Flink de forma local con el objetivo a largo plazo de implementarlas en Managed Service para Apache Flink.

La solución se encuentra aquí: [Guía para principiantes sobre el desarrollo local con Apache Flink](#)

Detección de eventos con Managed Service para Apache Flink Studio

En este taller, se describe la detección de eventos con Managed Service para Apache Flink Studio y su implementación como una aplicación de Managed Service para Apache Flink

La solución se encuentra aquí: [Detección de eventos con Managed Service para Apache Flink](#)

Solución de datos de streaming de AWS para Amazon Kinesis

La solución de datos de streaming de AWS para Amazon Kinesis configura automáticamente los servicios de AWS necesarios para capturar, almacenar, procesar y entregar fácilmente los datos de streaming. La solución ofrece varias opciones para resolver casos de uso de datos de streaming. La opción Managed Service for Apache Flink proporciona un ejemplo de ETL end-to-end en streaming que muestra una aplicación real que ejecuta operaciones analíticas con datos simulados de taxis de Nueva York.

Cada solución incluye los siguientes componentes:

- Un paquete de AWS CloudFormation para implementar el ejemplo completo.
- Un CloudWatch panel de control para mostrar las métricas de la aplicación.
- CloudWatch alarmas sobre las métricas de aplicación más relevantes.
- Todas las políticas y roles de IAM necesarios.

La solución se encuentra aquí: [Solución de datos de streaming para Amazon Kinesis](#)

Clickstream Lab con Apache Flink y Apache Kafka

Un laboratorio integral para casos de uso de secuencias de clics que utiliza Amazon Managed Streaming para Apache Kafka para almacenamiento de streaming y aplicaciones Managed Service para Apache Flink para procesamiento de flujos.

La solución se encuentra aquí: [Clickstream Lab](#)

Escalado personalizado utilizando el escalado automático de aplicaciones

Un ejemplo que ayuda a los usuarios a escalar automáticamente sus aplicaciones de Managed Service for Apache Flink mediante Application Auto Scaling. Esto permite a los usuarios configurar políticas de escalado personalizadas y atributos de escalado personalizados.

Las soluciones se encuentran aquí:

- [Servicio gestionado para el escalado automático de la aplicación Apache Flink](#)
- [Escalado programado](#)

Para obtener más información sobre cómo realizar un escalado personalizado, consulte [Habilitar el escalado programado y basado en métricas para Amazon Managed Service for Apache Flink](#).

CloudWatch Panel de control de Amazon

Un ejemplo de CloudWatch panel para monitorear el servicio administrado para las aplicaciones de Apache Flink. El panel de ejemplo también incluye una [aplicación de demostración](#) para ayudar a demostrar la funcionalidad del panel.

La solución se encuentra aquí: [Panel de métricas de Managed Service para Apache Flink](#)

Solución de datos de streaming de AWS para Amazon MSK

La solución de datos de streaming de AWS para Amazon MSK proporciona plantillas de AWS CloudFormation en las que los datos fluyen a través de los productores, el almacenamiento de streaming, los consumidores y los destinos.

La solución se encuentra aquí: [AWS Solución de datos de streaming para Amazon MSK](#)

Más servicios gestionados para Apache Flink Solutions en GitHub

Los siguientes end-to-end ejemplos muestran las soluciones avanzadas de servicios gestionados para Apache Flink y están disponibles en: GitHub

- [Amazon Managed Service para Apache Flink: utilidad de evaluación comparativa](#)
- [Gestor de instantáneas: Amazon Managed Service para Apache Flink](#)
- [ETL de streaming con Apache Flink y Amazon Managed Service para Apache Flink](#)
- [Análisis de opiniones de los clientes en tiempo real](#)

Utilidades

Las siguientes utilidades pueden facilitar el uso del servicio Managed Service para Apache Flink:

Temas

- [Gestor de instantáneas](#)
- [Evaluación comparativa](#)

Gestor de instantáneas

Se recomienda que las aplicaciones de Flink activen puntos de guardado o instantáneas con regularidad para permitir una recuperación de fallos más fluida. El gestor de instantáneas automatiza esta tarea y ofrece los siguientes beneficios:

- toma una nueva instantánea de una aplicación de Managed Service para Apache Flink en ejecución
- obtiene un recuento de las instantáneas de las aplicaciones
- comprueba si el recuento es superior al número requerido de instantáneas
- elimina las instantáneas más antiguas que el número requerido

Para ver un ejemplo, consulte [Gestor de instantáneas](#).

Evaluación comparativa

la utilidad de evaluación comparativa de Managed Service para Apache Flink ayuda en la planificación de la capacidad, las pruebas de integración y la evaluación comparativa de aplicaciones de Managed Service para Apache Flink.

Para ver un ejemplo, consulte [Evaluación comparativa](#)

Managed Service para Apache Flink: ejemplos

En esta sección se proporcionan ejemplos de cómo crear y utilizar aplicaciones en Managed Service para Apache Flink. Incluyen ejemplos de código e step-by-step instrucciones para ayudarle a crear un servicio gestionado para las aplicaciones de Apache Flink y comprobar sus resultados.

Antes de explorar estos ejemplos, le recomendamos que en primer lugar examine lo siguiente:

- [Cómo funciona](#)
- [Cómo empezar \(DataStream API\)](#)

Note

En estos ejemplos se asume que utiliza la región Oeste de EE. UU. (Oregón) (us-west-2). Si utiliza una región diferente, actualice el código de la aplicación, los comandos y los roles de IAM en concordancia.

Temas

- [DataStream Ejemplos de API](#)
- [Ejemplos de Python](#)
- [Ejemplo de Scala](#)

DataStream Ejemplos de API

Los siguientes ejemplos muestran cómo crear aplicaciones mediante la DataStream API Apache Flink.

Temas

- [Ejemplo: ventana de salto de tamaño constante](#)
- [Ejemplo: ventana deslizante](#)
- [Ejemplo: escribir a un bucket de Amazon S3](#)
- [Tutorial: Uso de un servicio gestionado para la aplicación Apache Flink para replicar datos de un tema de un clúster de MSK a otro en una VPC](#)
- [Ejemplo: utilizar un consumidor EFO con un flujo de datos de Kinesis](#)

- [Ejemplo: Escritura en Kinesis Data Firehose](#)
- [Ejemplo: leer desde un flujo de Kinesis en una cuenta diferente.](#)
- [Tutorial: Cómo usar un Truststore personalizado con Amazon MSK](#)

Ejemplo: ventana de salto de tamaño constante

En este ejercicio, creará una aplicación de Managed Service para Apache Flink que agrega datos mediante una ventana de salto de tamaño constante. La agregación se encuentra habilitada de manera predeterminada en Flink. Para deshabilitarla, utilice lo siguiente:

```
sink.producer.aggregation-enabled' = 'false'
```

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de datos de Kinesis (ExampleInputStream y ExampleOutputStream)
- Un bucket de Amazon S3 para almacenar el código de la aplicación (ka-app-code-*<username>*)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a los flujos de datos **ExampleInputStream** y **ExampleOutputStream**.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
```

```
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Java para este ejemplo está disponible en [GitHub](#). Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/TumblingWindow`.

El código de la aplicación se encuentra en el archivo `TumblingWindowStreamingJob.java`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
        new SimpleStringSchema(), inputProperties));
```

- Añada la siguiente instrucción import:

```
import
  org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- La aplicación utiliza el operador `timeWindow` para encontrar el recuento de valores de cada símbolo bursátil en una ventana de salto de tamaño constante de 5 segundos. El siguiente código crea el operador y envía los datos agregados a un nuevo receptor de flujo de datos de Kinesis:

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
      .keyBy(0) // Logically partition the stream for each word

      .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //
Flink 1.13 onward
      .sum(1) // Sum the number of words per partition
      .map(value -> value.f0 + "," + value.f1.toString() + "\n")
      .addSink(createSinkFromStaticConfig());
```

Compilar el código de la aplicación

Para compilar la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale Java y Maven. Para obtener más información, consulte [Requisitos previos](#) en el tutorial de [Cómo empezar \(DataStream API\)](#).
2. Compile la aplicación con el siguiente comando:

```
mvn package -Dflink.version=1.15.3
```

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Al compilar la aplicación, se crea el archivo JAR de la aplicación (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Cargar el código de Java de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. En la consola de Amazon S3, elija el <username>bucket ka-app-code- y, a continuación, seleccione Upload.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la aplicación, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink.

Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-apps-1.0.jar"
      ]
    }
  ],
}
```



```

    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": "logs:DescribeLogStreams",
      "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
  ]
}

```

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
5. Para el CloudWatch registro, active la casilla Activar.
6. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Ejecutar la aplicación

1. En la MyApplication página, seleccione Ejecutar. Deje seleccionada la opción Ejecutar sin instantánea y confirme la acción.
2. Cuando la aplicación se está ejecutando, actualice la página. La consola muestra el Gráfico de la aplicación.

Puede comprobar las métricas del servicio gestionado para Apache Flink en la CloudWatch consola para comprobar que la aplicación funciona.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana de salto de tamaño constante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija. MyApplication
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: ventana deslizante

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)

- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`).
- Un bucket de Amazon S3 para almacenar el código de la aplicación (`ka-app-code-<username>`)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a sus flujos de datos **ExampleInputStream** y **ExampleOutputStream**.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3
```

```
STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
            PartitionKey="partitionkey"
        )

if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Java de este ejemplo está disponible en [GitHub](#). Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/SlidingWindow`.

El código de la aplicación se encuentra en el archivo

`SlidingWindowStreamingJobWithParallelism.java`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
        new SimpleStringSchema(), inputProperties));
```

- La aplicación utiliza al operador `timeWindow` para encontrar el valor mínimo para cada símbolo bursátil en una ventana de 10 segundos que se desliza 5 segundos. El siguiente código crea el operador y envía los datos agregados a un nuevo receptor de flujo de datos de Kinesis:
- Añada la siguiente instrucción `import`:

```
import
    org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
    flink 1.13 onward
```

- La aplicación utiliza el operador `timeWindow` para encontrar el recuento de valores de cada símbolo bursátil en una ventana de salto de tamaño constante de 5 segundos. El siguiente código crea el operador y envía los datos agregados a un nuevo receptor de flujo de datos de Kinesis:

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
        .keyBy(0) // Logically partition the stream for each word

        .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink 1.13 onward
        .sum(1) // Sum the number of words per partition
        .map(value -> value.f0 + "," + value.f1.toString() + "\n")
        .addSink(createSinkFromStaticConfig());
```

Compilar el código de la aplicación

Para compilar la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale Java y Maven. Para obtener más información, consulte [Requisitos previos](#) en el tutorial de [Cómo empezar \(DataStream API\)](#).

2. Compile la aplicación con el siguiente comando:

```
mvn package -Dflink.version=1.15.3
```

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Al compilar la aplicación, se crea el archivo JAR de la aplicación (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Cargar el código de Java de streaming de Apache Flink

En esta sección, carga el código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. En la consola de Amazon S3, elija el `<username>bucket ka-app-code-` y, a continuación, elija Upload.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la aplicación, escriba **MyApplication**.

- En Tiempo de ejecución, escriba Apache Flink.
 - Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

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

```

        "s3:GetObject",
        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
    ]
},
{
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": "logs:DescribeLogStreams",
    "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
},
{
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": "logs:PutLogEvents",
    "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
},
{
    "Sid": "ListCloudwatchLogGroups",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
},
{
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
},
{
    "Sid": "WriteOutputStream",
    "Effect": "Allow",

```

```
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
]
}
```

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
5. Para el CloudWatch registro, active la casilla Activar.
6. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: /aws/kinesis-analytics/MyApplication
- Flujo de registro: kinesis-analytics-log-stream

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Configure el paralelismo de la aplicación

En este ejemplo de aplicación se utiliza la ejecución paralela de tareas. El siguiente código de aplicación establece el paralelismo del operador `min`:

```
.setParallelism(3) // Set parallelism for the min operator
```

El paralelismo de la aplicación no puede ser mayor que el paralelismo provisto, que tiene un valor predeterminado de 1. Para aumentar el paralelismo de la aplicación, realice la siguiente acción AWS CLI:

```
aws kinesisanalyticstv2 update-application
  --application-name MyApplication
  --current-application-version-id <VersionId>
  --application-configuration-update "{\"FlinkApplicationConfigurationUpdate
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
  \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

Puede recuperar el ID de la versión actual de la aplicación mediante las [ListApplications](#) acciones [DescribeApplication](#).

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Puede comprobar las métricas del servicio gestionado para Apache Flink en la CloudWatch consola para comprobar que la aplicación funciona.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana deslizable.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)

- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Servicio gestionado para Apache Flink, elija. MyApplication
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStream página, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: escribir a un bucket de Amazon S3

En este ejercicio, usted deberá crear un Managed Service para Apache Flink con un flujo de datos de Kinesis como origen y un bucket de Amazon S3 como receptor. Con el receptor, puede verificar la salida de la aplicación en la consola de Amazon S3.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Modificar el código de la aplicación](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Verificar la salida de la aplicación](#)
- [Opcional: personalice el origen y el receptor](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear un Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Un flujo de datos de Kinesis (`ExampleInputStream`).
- Un bucket de Amazon S3 para almacenar el código y los resultados de la aplicación (`ka-app-code-<username>`)

Note

Managed Service para Apache Flink no puede escribir datos en Amazon S3 con el cifrado del lado del servidor activado en Managed Service para Apache Flink.

Puede crear el flujo de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne el nombre **ExampleInputStream** a su flujo de datos.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***. Cree dos carpetas (**code** y **data**) en el bucket de Amazon S3.

La aplicación crea los siguientes recursos si aún no existen: CloudWatch

- Un grupo de registro llamado `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flujo de registro llamado `kinesis-analytics-log-stream`.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/S3Sink`.

El código de la aplicación se encuentra en el archivo `S3StreamingSinkJob.java`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- Debe añadir la siguiente instrucción `import`:

```
import  
    org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- La aplicación utiliza un receptor Apache Flink S3 para escribir en Amazon S3.

El receptor lee los mensajes en una ventana de saltos de tamaño constante, los codifica en objetos de bucket de S3 y envía los objetos codificados al receptor de S3. El siguiente código codifica los objetos para enviarlos a Amazon S3:

```
input.map(value -> { // Parse the JSON  
    JsonNode jsonNode = jsonParser.readValue(value, JsonNode.class);  
    return new Tuple2<>(jsonNode.get("ticker").toString(), 1);  
}).returns(Types.TUPLE(Types.STRING, Types.INT))  
    .keyBy(v -> v.f0) // Logically partition the stream for each word  
    .window(TumblingProcessingTimeWindows.of(Time.minutes(1)))  
    .sum(1) // Count the appearances by ticker per partition  
    .map(value -> value.f0 + " count: " + value.f1.toString() + "\n")  
    .addSink(createS3SinkFromStaticConfig());
```

Note

La aplicación utiliza un objeto Flink `StreamingFileSink` para escribir en Amazon S3. Para obtener más información sobre `StreamingFileSink`, consulte [StreamingFileSink](#) la [documentación de Apache Flink](#).

Modificar el código de la aplicación

En esta sección, modificará el código de la aplicación para escribir el resultado en su bucket de Amazon S3.

Actualice la siguiente línea con su nombre de usuario para especificar la ubicación de salida de la aplicación:

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

Compilar el código de la aplicación

Para compilar la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale Java y Maven. Para obtener más información, consulte [Requisitos previos](#) en el tutorial de [Cómo empezar \(DataStream API\)](#).
2. Compile la aplicación con el siguiente comando:

```
mvn package -Dflink.version=1.15.3
```

Al compilar la aplicación, se crea el archivo JAR de la aplicación (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Cargar el código de Java de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. En la consola de Amazon S3, elija el <username>bucket ka-app-code-, vaya a la carpeta de códigos y elija Upload.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la aplicación, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La

aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- En Nombre de la aplicación, escriba **MyApplication**.
- En Tiempo de ejecución, escriba Apache Flink.
- Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).

6. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
7. Elija Crear aplicación.

Note

Al crear un Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso al flujo de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta. Sustituya `<username>` por el nombre de usuario.

```
{  
    "Sid": "S3",
```

```

    "Effect": "Allow",
    "Action": [
      "s3:Abort*",
      "s3:DeleteObject*",
      "s3:GetObject*",
      "s3:GetBucket*",
      "s3:List*",
      "s3:ListBucket",
      "s3:PutObject"
    ],
    "Resource": [
      "arn:aws:s3:::ka-app-code-<username>",
      "arn:aws:s3:::ka-app-code-<username>/*"
    ]
  },
  {
    "Sid": "ListCloudwatchLogGroups",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:region:account-id:log-group:*"
    ]
  },
  {
    "Sid": "ListCloudwatchLogStreams",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER
%:log-stream:*"
    ]
  },
  {
    "Sid": "PutCloudwatchLogs",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [

```

```
        "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER
%:log-stream:%LOG_STREAM_PLACEHOLDER%"
    ]
}
',
{
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
},
]
}
```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **code/aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
5. Para el CloudWatch registro, active la casilla Activar.
6. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`

- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Ejecutar la aplicación

1. En la MyApplication página, seleccione Ejecutar. Deje seleccionada la opción Ejecutar sin instantánea y confirme la acción.
2. Cuando la aplicación se está ejecutando, actualice la página. La consola muestra el Gráfico de la aplicación.

Verificar la salida de la aplicación

En la consola de Amazon S3, abra la carpeta de datos de su bucket de S3.

Transcurridos unos minutos, aparecerán los objetos que contienen datos agregados de la aplicación.

Note

La agregación se encuentra habilitada de manera predeterminada en Flink. Para deshabilitarla, utilice lo siguiente:

```
sink.producer.aggregation-enabled' = 'false'
```

Opcional: personalice el origen y el receptor

En esta sección, personalizará la configuración de los objetos origen y receptor.

Note

Tras cambiar las secciones de código descritas en las secciones siguientes, haga lo siguiente para volver a cargar el código de la aplicación:

- Repita los pasos de la sección [the section called “Compilar el código de la aplicación”](#) para compilar el código de la aplicación actualizado.

- Repita los pasos de la sección [the section called “Cargar el código de Java de streaming de Apache Flink”](#) para cargar el código de la aplicación actualizado.
- En la página de la aplicación en la consola, seleccione Configurar y, a continuación, seleccione Actualizar para volver a cargar en la aplicación el código de la aplicación actualizado.

Esta sección contiene lo siguiente:

- [Configurar la partición de datos](#)
- [Configurar frecuencia de lectura](#)
- [Configurar almacenamiento en búfer de escritura](#)

Configurar la partición de datos

En esta sección, configurará los nombres de las carpetas que el receptor de archivos de streaming crea en el bucket de S3. Para ello, añada un asignador de bucket al receptor de archivos de streaming.

Para personalizar los nombres de carpetas creados en el bucket de S3, haga lo siguiente:

1. Añada las siguientes instrucciones de importación al principio del archivo `S3StreamingSinkJob.java`:

```
import
  org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPolicy;
import
  org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAssigner;
```

2. Actualice el método `createS3SinkFromStaticConfig()` en el código para que tenga el siguiente aspecto:

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

    final StreamingFileSink<String> sink = StreamingFileSink
        .forRowFormat(new Path(s3SinkPath), new
        SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(DefaultRollingPolicy.create().build())
}
```



```
        .build();
    return sink;
}
```

El ejemplo de código anterior utiliza el `DateTimeBucketAssigner` con un formato de fecha personalizado para crear carpetas en el bucket de S3. El `DateTimeBucketAssigner` utiliza la hora actual del sistema para crear los nombres de los buckets. Si quieres crear un asignador de cubos personalizado para personalizar aún más los nombres de las carpetas creadas, puedes crear una clase que lo implemente [BucketAssigner](#). Para implementar la lógica personalizada, utilice el método `getBucketId`.

Una implementación personalizada de `BucketAssigner` puede usar el parámetro [Contexto](#) para obtener más información sobre un registro con el fin de determinar su carpeta de destino.

Configurar frecuencia de lectura

En esta sección, se configura la frecuencia de las lecturas en la transmisión de origen.

De forma predeterminada, el consumidor de flujos de Kinesis lee la transmisión de origen cinco veces por segundo. Esta frecuencia provocará problemas si hay más de un cliente leyendo la transmisión o si la aplicación necesita volver a intentar leer un registro. Puede evitar estos problemas configurando la frecuencia de lectura del consumidor.

Para establecer la frecuencia de lectura del consumidor de Kinesis, debe configurar el parámetro `SHARD_GETRECORDS_INTERVAL_MILLIS`.

El siguiente ejemplo de código establece el parámetro `SHARD_GETRECORDS_INTERVAL_MILLIS` en un segundo:

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS, "1000");
```

Configurar almacenamiento en búfer de escritura

En esta sección, configurará la frecuencia de escritura y otros ajustes del receptor.

De forma predeterminada, la aplicación escribe en el bucket de destino cada minuto. Puede cambiar este intervalo y otros ajustes configurando el objeto `DefaultRollingPolicy`.

Note

El receptor de archivos de streaming de Apache Flink escribe en su bucket de salida cada vez que la aplicación crea un punto de control. De forma predeterminada, la aplicación crea un punto de control cada minuto. Para aumentar el intervalo de escritura del receptor S3, también debe aumentar el intervalo del punto de control.

Para configurar el objeto `DefaultRollingPolicy`, haga lo siguiente:

1. Aumente el parámetro `CheckpointInterval` de la aplicación. La siguiente entrada para la [UpdateApplication](#) acción establece el intervalo del punto de control en 10 minutos:

```
{
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "CheckpointConfigurationUpdate": {
        "ConfigurationTypeUpdate" : "CUSTOM",
        "CheckpointIntervalUpdate": 600000
      }
    }
  },
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5
}
```

Para utilizar el código anterior, especifique la versión actual de la aplicación. Puede recuperar la versión de la aplicación mediante la [ListApplications](#) acción.

2. Añada la siguiente instrucción de importación al principio del archivo `S3StreamingSinkJob.java`:

```
import java.util.concurrent.TimeUnit;
```

3. Actualice el método `createS3SinkFromStaticConfig` en el archivo `S3StreamingSinkJob.java` para que tenga el siguiente aspecto:

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {
    final StreamingFileSink<String> sink = StreamingFileSink
```

```
        .forRowFormat(new Path(s3SinkPath), new
SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(
            DefaultRollingPolicy.create()
                .withRolloverInterval(TimeUnit.MINUTES.toMillis(8))
                .withInactivityInterval(TimeUnit.MINUTES.toMillis(5))
                .withMaxPartSize(1024 * 1024 * 1024)
                .build())
        .build();
    return sink;
}
```

El ejemplo de código anterior establece la frecuencia de escrituras en el bucket de Amazon S3 en 8 minutos.

Para obtener más información sobre la configuración del receptor de archivos de streaming de Apache Flink, consulte [Formatos codificados por filas](#) en la [documentación de Apache Flink](#).

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial de Amazon S3.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar su flujo de datos de Kinesis](#)
- [Eliminar objetos y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar su flujo de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.

Eliminar objetos y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Tutorial: Uso de un servicio gestionado para la aplicación Apache Flink para replicar datos de un tema de un clúster de MSK a otro en una VPC

El siguiente tutorial muestra cómo crear una Amazon VPC con un clúster de Amazon MSK y dos temas, y cómo crear una aplicación de servicio gestionado para Apache Flink que lee desde un tema de Amazon MSK y escribe en otro.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Este tutorial contiene las siguientes secciones:

- [Crear un Amazon VPC con un clúster de Amazon MSK](#)
- [Crear el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear la aplicación](#)
- [Configurar la aplicación](#)
- [Ejecutar la aplicación](#)
- [Probar la aplicación](#)

Crear un Amazon VPC con un clúster de Amazon MSK

Para crear una muestra de VPC y un clúster de Amazon MSK al que acceder desde la aplicación de Managed Service para Apache Flink, siga el tutorial [Introducción al uso de Amazon MSK](#).

Cuando complete el tutorial, tenga en cuenta lo siguiente:

- En el [Paso 3: Crear un tema](#), repita el comando `kafka-topics.sh --create` para crear un tema de destino llamado `AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

- Registre la lista de servidores de arranque de su clúster. Puede obtener la lista de servidores de arranque con el siguiente comando (sustitúyalo por `ClusterArn` el ARN de su clúster de MSK):

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
  "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Al seguir los pasos de los tutoriales, asegúrese de usar la región de AWS seleccionada en el código, los comandos y las entradas de la consola.

Crear el código de la aplicación

En esta sección, descargará y compilará el archivo JAR de la aplicación. Recomendamos utilizar Java 11.

El código de la aplicación Java de este ejemplo está disponible en [GitHub](#). Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. El código de la aplicación se encuentra en el archivo `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java`. Puede examinar el código para familiarizarse con la estructura del código de la aplicación Managed Service para Apache Flink.
4. Utilice la herramienta Maven de línea de comandos o el entorno de desarrollo que prefiera para crear el archivo JAR. Para compilar el archivo JAR utilizando la herramienta Maven de línea de comandos, introduzca lo siguiente:

```
mvn package -Dflink.version=1.15.3
```

Si la compilación es correcta, se crea el siguiente archivo:

```
target/KafkaGettingStartedJob-1.0.jar
```

Note

El código fuente proporcionado se basa en bibliotecas de Java 11. Si está utilizando un entorno de desarrollo,

Cargar el código de Java de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en el tutorial [Cómo empezar \(DataStream API\)](#).

Note

Si ha eliminado el bucket de Amazon S3 del tutorial de Introducción, vuelva a seguir el paso [the section called “Cargar el código de Java de streaming de Apache Flink”](#).

1. En la consola de Amazon S3, elija el <username>bucket ka-app-code- y, a continuación, seleccione Upload.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `KafkaGettingStartedJob-1.0.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink versión 1.15.2.

4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación


1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `KafkaGettingStartedJob-1.0.jar`.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM `kinesis-analytics-MyApplication-us-west-2`.

Note

Al especificar los recursos de la aplicación mediante la consola (como CloudWatch Logs o una Amazon VPC), la consola modifica la función de ejecución de la aplicación para conceder permiso de acceso a esos recursos.

4. En Propiedades, elija Añadir grupo. Introduzca las siguientes propiedades:

ID de grupo	Clave	Valor
KafkaSource	tema	AWSKafkaTutorialTopic
KafkaSource	bootstrap.servers	<i>La lista de servidores de arranque que guardó anteriormente</i>
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.contraseña	changeit

 Note

El `ssl.truststore.password` del certificado predeterminado es "changeit"; no necesita cambiar este valor si utiliza el certificado predeterminado.

Vuelva a seleccionar Añadir grupo. Introduzca las siguientes propiedades:

ID de grupo	Clave	Valor
KafkaSink	tema	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	<i>La lista de servidores de arranque que guardó anteriormente</i>
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts

ID de grupo	Clave	Valor
KafkaSink	ssl.truststore.contraseña	changeit
KafkaSink	transacción.timeout.ms	1 000

El código de la aplicación lee las propiedades de la aplicación anteriores para configurar el origen y el receptor que se utilizan para interactuar con la VPC y el clúster de Amazon MSK. Para obtener más información acerca de las propiedades, consulte [Propiedades de tiempo de ejecución](#).

5. En Instantáneas, seleccione Desactivar. Esto facilitará la actualización de la aplicación sin cargar datos de estado de la aplicación no válidos.
6. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
7. Para el CloudWatch registro, seleccione la casilla Activar.
8. En la sección Nube privada virtual (VPC), elija la VPC que desea asociar a la aplicación. Elija las subredes y el grupo de seguridad asociados a la VPC que desee que utilice la aplicación para acceder a los recursos de la VPC.
9. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación.

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Probar la aplicación

En esta sección, se escriben los registros en el tema de origen. La aplicación lee los registros del tema de origen y los escribe en el tema de destino. Para comprobar que la aplicación funciona, escriba los registros en el tema de origen y lea los registros del tema de destino.

Para escribir y leer registros de los temas, siga los pasos del [Paso 6: Producir y consumir datos](#) del tutorial [Cómo empezar a utilizar Amazon MSK](#).

Para leer el tema de destino, utilice el nombre del tema de destino en lugar del tema de origen en la segunda conexión al clúster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --  
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-  
beginning
```

Si no aparece ningún registro en el tema de destino, consulte la sección [No puede obtener acceso a los recursos de una VPC](#) del tema [Solución de problemas](#).

Ejemplo: utilizar un consumidor EFO con un flujo de datos de Kinesis

En este ejercicio, creará una aplicación de servicio gestionado para Apache Flink que lea desde un flujo de datos de Kinesis mediante [un consumidor de fan-out mejorado \(Enhanced Fan-Out, EFO\)](#). Si un consumidor de Kinesis usa EFO, el servicio Kinesis Data Streams le proporciona su propio ancho de banda dedicado, en lugar de hacer que el consumidor comparta el ancho de banda fijo del flujo con los demás consumidores que leen el flujo.

Para obtener más información sobre el uso de EFO con el consumidor Kinesis, consulte [FLIP-128: distribución mejorada para consumidores de Kinesis](#).

La aplicación que crea en este ejemplo utiliza AWS Kinesis Connector (flink-connector-kinesis) 1.15.3.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`)
- Un bucket de Amazon S3 para almacenar el código de la aplicación (`ka-app-code-<username>`)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a los flujos de datos **ExampleInputStream** y **ExampleOutputStream**.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Java de este ejemplo está disponible en [GitHub](#). Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/EfoConsumer`.

El código de la aplicación se encuentra en el archivo `EfoApplication.java`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Para activar el consumidor EFO, configure los siguientes parámetros en el consumidor de Kinesis:
 - `RECORD_PUBLISHER_TYPE`: defina este parámetro en EFO para que su aplicación utilice un consumidor de EFO para acceder a los datos del flujo de datos de Kinesis.
 - `EFO_CONSUMER_NAME`: defina este parámetro en un valor de cadena que sea único entre los consumidores de este flujo. La reutilización de un nombre de consumidor en el mismo flujo de datos de Kinesis provocará la cancelación del consumidor anterior que utilizó ese nombre.
- El siguiente ejemplo de código muestra cómo asignar valores a las propiedades de configuración del consumidor para utilizar un consumidor EFO para leer la transmisión de origen:


```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");  
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

Compilar el código de la aplicación

Para compilar la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale Java y Maven. Para obtener más información, consulte [Requisitos previos](#) en el tutorial de [Cómo empezar \(DataStream API\)](#).
2. Compile la aplicación con el siguiente comando:

```
mvn package -Dflink.version=1.15.3
```

 Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Al compilar la aplicación, se crea el archivo JAR de la aplicación (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Cargar el código de Java de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. En la consola de Amazon S3, elija el `<username>bucket ka-app-code-` y, a continuación, seleccione Upload.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.


Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación


1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:

- En Nombre de la aplicación, escriba **MyApplication**.
- En Tiempo de ejecución, escriba Apache Flink.

 Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

 Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

Note

Estos permisos otorgan a la aplicación la posibilidad de acceder al consumidor de EFO.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-apps-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": "logs:DescribeLogStreams",
      "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/MyApplication:log-stream:*"
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
    }
  ]
}
```

```

    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
  },
  {
    "Sid": "AllStreams",
    "Effect": "Allow",
    "Action": [
      "kinesis:ListShards",
      "kinesis:ListStreamConsumers",
      "kinesis:DescribeStreamSummary"
    ],
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/*"
  },
  {
    "Sid": "Stream",
    "Effect": "Allow",
    "Action": [
      "kinesis:DescribeStream",
      "kinesis:RegisterStreamConsumer",
      "kinesis:DeregisterStreamConsumer"
    ],
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  },
  {
    "Sid": "Consumer",
    "Effect": "Allow",
    "Action": [
      "kinesis:DescribeStreamConsumer",
      "kinesis:SubscribeToShard"
    ],
    "Resource": [
      "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app",
      "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app:*"
    ]
  }
}

```

```

    ]
  }
}

```

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, elija Crear grupo.
5. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
ConsumerConfigProperties	flink.stream.recorderpublisher	EFO
ConsumerConfigProperties	flink.stream.efo.consumername	basic-efo-flink-app
ConsumerConfigProperties	INPUT_STREAM	ExampleInputStream
ConsumerConfigProperties	flink.inputstream.initpos	LATEST
ConsumerConfigProperties	AWS_REGION	us-west-2

6. En Propiedades, elija Crear grupo.
7. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
ProducerConfigProperties	OUTPUT_STREAM	ExampleOutputStream
ProducerConfigProperties	AWS_REGION	us-west-2
ProducerConfigProperties	AggregationEnabled	false

8. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
9. Para el CloudWatch registro, active la casilla Activar.
10. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Puede comprobar las métricas del servicio gestionado para Apache Flink en la CloudWatch consola para comprobar que la aplicación funciona.

También puede consultar el nombre de su consumidor `basic-efo-flink-app()` en la consola de Kinesis Data Streams, en la pestaña desplegable mejorada de la transmisión de datos.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana efo.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija. MyApplication
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: Escritura en Kinesis Data Firehose

En este ejercicio, creará una aplicación de Managed Service para Apache Flink que tenga un flujo de datos de Kinesis como fuente y un flujo de Kinesis Data Firehose como receptor. Con el receptor, puede verificar la salida de la aplicación en un bucket de Amazon S3.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

Esta sección contiene los siguientes pasos:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)

- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear un Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Un flujo de datos de Kinesis (`ExampleInputStream`)
- Un flujo de Kinesis Data Firehose en el que la aplicación escribe los datos de salida a (`ExampleDeliveryStream`).
- Un bucket de Amazon S3 para almacenar el código de la aplicación (`ka-app-code-<username>`)

Puede crear el flujo de Kinesis, los buckets de Amazon S3 y el flujo de Kinesis Data Firehose mediante la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne el nombre **ExampleInputStream** a su flujo de datos.
- [Creación de un flujo de entrega de Amazon Kinesis Data Firehose](#) en la guía para desarrolladores de Amazon Kinesis Data Firehose. Otorgue un nombre a su flujo de Kinesis Data Firehose **ExampleDeliveryStream**. Al crear el flujo de Kinesis Data Firehose, cree también el destino S3 y el rol de IAM de dicho flujo.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java de este ejemplo está disponible en [GitHub](#). Para descargar el código de la aplicación, haga lo siguiente:

1. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/FirehoseSink`.

El código de la aplicación se encuentra en el archivo `FirehoseSinkStreamingJob.java`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- La aplicación utiliza un receptor Kinesis Data Firehose para escribir datos en un flujo de Kinesis Data Firehose. El siguiente fragmento crea el receptor de Kinesis Data Firehose:

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {  
    Properties sinkProperties = new Properties();  
    sinkProperties.setProperty(AWS_REGION, region);  
  
    return KinesisFirehoseSink.<String>builder()  
        .setFirehoseClientProperties(sinkProperties)  
        .setSerializationSchema(new SimpleStringSchema())  
        .setDeliveryStreamName(outputDeliveryStreamName)  
        .build();  
}
```


Compilar el código de la aplicación

Para compilar la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale Java y Maven. Para obtener más información, consulte [Requisitos previos](#) en el tutorial de [Cómo empezar \(DataStream API\)](#).

2. Para utilizar el conector de Kinesis para la siguiente aplicación, debe descargar, compilar e instalar Apache Maven. Para obtener más información, consulte [the section called “Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink”](#).
3. Compile la aplicación con el siguiente comando:

```
mvn package -Dflink.version=1.15.3
```

 Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Al compilar la aplicación, se crea el archivo JAR de la aplicación (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Cargar el código de Java de streaming de Apache Flink

En esta sección, carga el código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. En la consola, selecciona el compartimento `ka-app-code- <username>` y, a continuación, selecciona Cargar.
3. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `java-getting-started-1.0.jar` que creó en el paso anterior.
4. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Puede crear y ejecutar una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI.

Note

Cuando crea la aplicación mediante la consola, sus recursos AWS Identity and Access Management (de IAM) y de Amazon CloudWatch Logs se crean automáticamente. Si crea la aplicación mediante la AWS CLI, debe crear estos recursos por separado.

Temas

- [Crear y ejecutar la aplicación \(consola\)](#)
- [Crear y ejecutar la aplicación \(AWS CLI\)](#)

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.

Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear la aplicación mediante la consola, tiene la opción de crear un rol de IAM y una política para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso al flujo de datos de Kinesis y el flujo de Kinesis Data Firehose.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace todas las instancias del ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
```

```

    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
  },
  {
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteDeliveryStream",
    "Effect": "Allow",
    "Action": "firehose:*",
    "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
  }
]

```

```
}
```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **java-getting-started-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
5. Para el CloudWatch registro, active la casilla Activar.
6. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

En la MyApplication página, seleccione Detener. Confirme la acción.

Actualizar la aplicación

Mediante la consola, puede actualizar la configuración de la aplicación, tal como sus propiedades, ajustes de monitorización y la ubicación o el nombre de archivo JAR de la aplicación.

En la MyApplication página, elija Configurar. Actualice la configuración de la aplicación y elija Actualizar.

Note

Para actualizar el código de la aplicación en la consola, debe cambiar el nombre del objeto del JAR, usar un bucket de S3 diferente o usar el AWS CLI como se describe en la sección [the section called “Actualizar el código de la aplicación”](#). Si el nombre del archivo o el bucket no cambian, el código de la aplicación no se vuelve a cargar al seleccionar Actualizar en la página de Configuración.

Crear y ejecutar la aplicación (AWS CLI)

En esta sección, se utiliza AWS CLI para crear y ejecutar la aplicación Managed Service para Apache Flink.

Crear una política de permisos

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción `read` en el flujo de origen y otra que concede permisos para las acciones `write` en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

`AKReadSourceStreamWriteSinkStream`. Reemplace el *nombre de usuario* por el nombre de usuario que utilizará para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Sustituya el ID de la cuenta en los nombres de recurso de Amazon (ARN) (*012345678901*) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
```

```

    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "Resource": ["arn:aws:s3:::ka-app-code-username",
      "arn:aws:s3:::ka-app-code-username/*"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteDeliveryStream",
    "Effect": "Allow",
    "Action": "firehose:*",
    "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
  }
]
}

```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial sobre cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Note

Para acceder a otros servicios de Amazon, puede usar AWS SDK for Java. Managed Service para Apache Flink establece automáticamente las credenciales requeridas por el SDK con las del rol de IAM de ejecución del servicio asociada a su aplicación. No hace falta realizar ningún otro paso.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su transmisión si no tiene permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza otorga permiso a Managed Service para Apache Flink para asumir el rol. La política de permisos determina qué puede hacer Managed Service para Apache Flink después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM


1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Roles, Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS. En Elegir el servicio que usará este rol, elija Kinesis. En Seleccionar su caso de uso, elija Kinesis Analytics.

Elija Siguiente: permisos.

4. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
5. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado MF-stream-rw-role. A continuación, actualice las políticas de confianza y permisos del rol.

6. Asocie la política de permisos al rol.

 Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Asocie la política que ha creado en el paso anterior, [the section called “Crear una política de permisos”](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la ReadSourceStreamWriteSinkStream política AK y elija Adjuntar política.

Ahora ha creado el rol de ejecución de servicio que utilizará la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación de Managed Service para Apache Flink

1. Copie el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket por el sufijo que eligió en la sección [the section called “Crear recursos dependientes”](#) (`ka-app-code-<username>`). Reemplace el ID de la cuenta de muestra (`012345678901`) en el rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "java-getting-started-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    }
  }
}
```

2. Ejecute la acción [CreateApplication](#) con la solicitud anterior para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{
  "ApplicationName": "test",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

2. Ejecute la acción [StartApplication](#) con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Ejecute la acción [StopApplication](#) con la siguiente solicitud para detener la aplicación:

```
aws kinesisanalyticstv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso CloudWatch de Logs con su aplicación, consulte [the section called “Configuración del registro”](#).

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción [UpdateApplication](#) de la AWS CLI.

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [the section called “Crear recursos dependientes”](#).

```
{
  "ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "java-getting-started-1.0.jar"
        }
      }
    }
  }
}
```

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para eliminar los recursos de AWS creados en el tutorial de Introducción.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar su flujo de datos de Kinesis](#)
- [Eliminar su flujo de Kinesis Data Firehose](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. Elija Configurar.
4. En la sección Instantáneas, seleccione Deshabilitar y, a continuación, seleccione Actualizar.
5. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar su flujo de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.

Eliminar su flujo de Kinesis Data Firehose

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Firehose, elija. ExampleDeliveryStream
3. En la ExampleDeliveryStreampágina, elija Eliminar transmisión de Kinesis Data Firehose y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. <username>Elija el cubo ka-app-code-.
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.
4. Si creó un bucket de Amazon S3 para el destino del flujo de Kinesis Data Firehose, elimínelo también.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. Si ha creado una nueva política para su flujo de Kinesis Data Firehose, elimínela también.
7. En la barra de navegación, seleccione Roles.
8. Elija la función kinesis-analytics - -us-west-2. MyApplication
9. Elija Eliminar rol y, a continuación, confirme la eliminación.
10. Si ha creado un rol nuevo para su flujo de Kinesis Data Firehose, elimínelo también.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: leer desde un flujo de Kinesis en una cuenta diferente.

En este ejemplo se muestra cómo crear una aplicación de servicio gestionado para Apache Flink que lea los datos de un flujo de Kinesis en una cuenta diferente. En este ejemplo, utilizará una cuenta

para el flujo de Kinesis de origen y una segunda cuenta para la aplicación Managed Service para Apache Flink y el flujo de receptor de Kinesis.

Este tema contiene las siguientes secciones:

- [Requisitos previos](#)
- [Configuración](#)
- [Crear flujo de origen de Kinesis](#)
- [Crear y actualizar las funciones y políticas de IAM](#)
- [Actualizar el script de Python](#)
- [Actualizar la aplicación Java](#)
- [Compile, cargue y ejecute la aplicación.](#)

Requisitos previos

- En este tutorial, modificará el ejemplo de introducción para leer los datos de un flujo de Kinesis en una cuenta diferente. Complete el tutorial [Cómo empezar \(DataStream API\)](#) antes de continuar.
- Necesita dos cuentas AWS para completar este tutorial: una para el flujo de origen y otra para el flujo de aplicación y del receptor. Utilice la cuenta AWS que utilizó en el tutorial de introducción para la aplicación y para el flujo del receptor. Use una cuenta AWS diferente para el flujo de origen.

Configuración

Accederá a sus dos cuentas AWS mediante perfiles con nombre. Modifique sus credenciales AWS y los archivos de configuración para incluir dos perfiles que contengan la información de región y conexión de sus dos cuentas.

El siguiente archivo de credenciales de ejemplo contiene dos perfiles con nombre, `ka-source-stream-account-profile` y `ka-sink-stream-account-profile`. Utilice la cuenta que utilizó en el tutorial de Introducción para la cuenta del flujo del receptor.

```
[ka-source-stream-account-profile]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

[ka-sink-stream-account-profile]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
```

```
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

El siguiente archivo de configuración de ejemplo contiene perfiles con el mismo nombre e información sobre la región y el formato de salida.

```
[profile ka-source-stream-account-profile]
region=us-west-2
output=json
```

```
[profile ka-sink-stream-account-profile]
region=us-west-2
output=json
```

Note

En este tutorial no se utiliza el `ka-sink-stream-account-profile`. Se incluye como ejemplo de cómo acceder a dos cuentas AWS diferentes mediante perfiles.

Para obtener más información sobre los perfiles con AWS CLI, consulte [Perfiles con nombre](#) en la documentación de AWS Command Line Interface.

Crear flujo de origen de Kinesis

En esta sección, creará el flujo de Kinesis en la cuenta de origen.

Ingrese el siguiente comando para crear el flujo de Kinesis que la aplicación utilizará como entrada. Tenga en cuenta que el parámetro `--profile` especifica qué perfil de cuenta utilizar.

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

Crear y actualizar las funciones y políticas de IAM

Para permitir el acceso a los objetos en todas las cuentas AWS, debe crear una política y un rol de IAM en la cuenta de origen. A continuación, modifique la política de IAM en la cuenta del receptor. Para obtener más información sobre cómo crear y administrar roles y políticas de IAM, consulte los siguientes temas en la AWS Identity and Access Management Guía del usuario:

- [Crear roles de IAM](#)
- [Crear políticas de IAM](#)

Funciones y políticas de la cuenta receptor

1. Edite la política `kinesis-analytics-service-MyApplication-us-west-2` desde el tutorial de Introducción. Esta política permite asumir el rol en la cuenta de origen para leer la transmisión de origen.

Note

Al usar la consola para crear la aplicación, la consola crea una política llamada `kinesis-analytics-service-<application name>-<application region>` y un rol denominado `kinesisanalytics-<application name>-<application region>`.

Añada a la política la sección resaltada a continuación. Sustituya el identificador de cuenta de ejemplo (`SOURCE01234567`) por el identificador de la cuenta que utilizará para el flujo de origen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleInSourceAccount",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::SOURCE01234567:role/KA-Source-Stream-Role"
    },
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3::ka-app-code-username/aws-kinesis-analytics-java-apps-1.0.jar"
      ]
    }
  ]
}
```

```

    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:SINK012345678:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
      ]
    }
  ]
}

```

- Abra el rol `kinesis-analytics-MyApplication-us-west-2` y anote su nombre de recurso de Amazon (ARN). Lo necesitará en la sección siguiente. El rol de ARN tiene el siguiente aspecto:

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-west-2
```

Funciones y políticas de la cuenta de origen

1. Crear una política en la cuenta de origen denominada `KA-Source-Stream-Policy`. Utilice el siguiente JSON para la política. Sustituya el número de cuenta de muestra por el número de cuenta de la cuenta de origen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetRecords",
        "kinesis:GetShardIterator",
        "kinesis:ListShards"
      ],
      "Resource":
        "arn:aws:kinesis:us-west-2:SOURCE123456784:stream/SourceAccountExampleInputStream"
    }
  ]
}
```

2. Crear un rol en la cuenta de origen denominada `MF-Source-Stream-Role`. Haga lo siguiente para crear el rol utilizando el caso de uso de Managed Flink:
 1. En la Consola de administración de IAM, seleccione Crear rol.
 2. En la página Crear función, elija `AWSServicio`. En la lista de servicios, elija Kinesis.
 3. En la sección Seleccione su caso de uso, elija Managed Service para Apache Flink.
 4. Elija Siguiente: permisos.
 5. Agregue la política de permisos `KA-Source-Stream-Policy` que creó en el paso anterior. Elija Siguiente:Etiquetas.
 6. Elija Siguiente:Revisar.
 7. Llame al rol `KA-Source-Stream-Role`. Su aplicación utilizará este rol para acceder al flujo de origen.
3. Agregue el ARN `kinesis-analytics-MyApplication-us-west-2` de la cuenta receptor a la relación de confianza del rol `KA-Source-Stream-Role` en la cuenta de origen:

1. Abra KA-Source-Stream-Role en la consola de IAM.
2. Seleccione la pestaña Relaciones de confianza.
3. Elija Editar relación de confianza.
4. Utilice el siguiente código para la relación de confianza. Reemplace el ID de la cuenta de muestra (*SINK012345678*) por el ID de su cuenta receptor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-west-2"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Actualizar el script de Python

En esta sección, actualizará el script de Python que genera datos de muestra para usar el perfil de la cuenta de origen.

Actualice el script `stock.py` con los siguientes cambios resaltados.

```
import json
import boto3
import random
import datetime
import os

os.environ['AWS_PROFILE'] = 'ka-source-stream-account-profile'
os.environ['AWS_DEFAULT_REGION'] = 'us-west-2'

kinesis = boto3.client('kinesis')
def getReferrer():
    data = {}
```

```

now = datetime.datetime.now()
str_now = now.isoformat()
data['event_time'] = str_now
data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
price = random.random() * 100
data['price'] = round(price, 2)
return data

while True:
    data = json.dumps(getReferrer())
    print(data)
    kinesis.put_record(
        StreamName="SourceAccountExampleInputStream",
        Data=data,
        PartitionKey="partitionkey")

```

Actualizar la aplicación Java

En esta sección, se actualiza el código de la aplicación Java para que asuma la función de cuenta de origen al leer el flujo de origen.

Realice los siguientes cambios en el archivo `BasicStreamingJob.java`. Sustituya el número de cuenta de origen del ejemplo (*SOURCE01234567*) por su número de cuenta de origen.

```

package com.amazonaws.services.managed-flink;

import com.amazonaws.services.managed-flink.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;
import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;
import org.apache.flink.streaming.connectors.kinesis.config.AWSConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

/**
 * A basic Managed Service for Apache Flink for Java application with Kinesis data
 * streams
 * as source and sink.

```

```
*/
public class BasicStreamingJob {
    private static final String region = "us-west-2";
    private static final String inputStreamName = "SourceAccountExampleInputStream";
    private static final String outputStreamName = ExampleOutputStream;
    private static final String roleArn = "arn:aws:iam::SOURCE01234567:role/KA-Source-Stream-Role";
    private static final String roleSessionName = "ksassumedrolesession";

    private static DataStream<String>
    createSourceFromStaticConfig(StreamExecutionEnvironment env) {
        Properties inputProperties = new Properties();
        inputProperties.setProperty(AWSConfigConstants.AWS_CREDENTIALS_PROVIDER,
            "ASSUME_ROLE");
        inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_ARN, roleArn);
        inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_SESSION_NAME,
            roleSessionName);
        inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
        inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
            "LATEST");

        return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
            SimpleStringSchema(), inputProperties));
    }

    private static KinesisStreamsSink<String> createSinkFromStaticConfig() {
        Properties outputProperties = new Properties();
        outputProperties.setProperty(AWSConfigConstants.AWS_REGION, region);

        return KinesisStreamsSink.<String>builder()
            .setKinesisClientProperties(outputProperties)
            .setSerializationSchema(new SimpleStringSchema())
            .setStreamName(outputProperties.getProperty("OUTPUT_STREAM",
                "ExampleOutputStream"))
            .setPartitionKeyGenerator(element ->
                String.valueOf(element.hashCode()))
            .build();
    }

    public static void main(String[] args) throws Exception {
        // set up the streaming execution environment
        final StreamExecutionEnvironment env =
            StreamExecutionEnvironment.getExecutionEnvironment();
    }
}
```

```
    DataStream<String> input = createSourceFromStaticConfig(env);

    input.addSink(createSinkFromStaticConfig());

    env.execute("Flink Streaming Java API Skeleton");
}
}
```

Compile, cargue y ejecute la aplicación.

Puede hacer lo siguiente para actualizar y ejecutar la aplicación:

1. Compile la aplicación nuevamente ejecutando el siguiente comando en el directorio con el archivo `pom.xml`.

```
mvn package -Dflink.version=1.15.3
```

2. Elimine el archivo JAR anterior del bucket de Amazon Simple Storage Service (Amazon S3) y, a continuación, cargue el archivo `aws-kinesis-analytics-java-apps-1.0.jar` nuevo en el bucket de S3.
3. En la página de la aplicación, en la consola de Managed Service para Apache Flink, seleccione Configurar y actualizar para volver a cargar el archivo JAR de la aplicación.
4. Ejecute el script `stock.py` para enviar los datos a la transmisión de origen.

```
python stock.py
```

La aplicación ahora lee los datos de la transmisión de Kinesis en la otra cuenta.

Puede comprobar que la aplicación funciona verificando la métrica de `PutRecords.Bytes` del flujo `ExampleOutputStream`. Si hay actividad en la transmisión de salida, la aplicación funciona correctamente.

Tutorial: Cómo usar un Truststore personalizado con Amazon MSK

API de origen de datos actual

Si utiliza las API de origen de datos actuales, su aplicación puede aprovechar la utilidad `MSK Config Providers` que se describe [aquí](#). Esto le permite a su `KafkaSource` función acceder al almacén de claves y al almacén de confianza para el TLS mutuo en Amazon S3.

```
...
// define names of config providers:
builder.setProperty("config.providers", "secretsmanager,s3import");

// provide implementation classes for each provider:
builder.setProperty("config.providers.secretsmanager.class",
    "com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider");
builder.setProperty("config.providers.s3import.class",
    "com.amazonaws.kafka.config.providers.S3ImportConfigProvider");

String region = appProperties.get(Helpers.S3_BUCKET_REGION_KEY).toString();
String keystoreS3Bucket = appProperties.get(Helpers.KEYSTORE_S3_BUCKET_KEY).toString();
String keystoreS3Path = appProperties.get(Helpers.KEYSTORE_S3_PATH_KEY).toString();
String truststoreS3Bucket =
    appProperties.get(Helpers.TRUSTSTORE_S3_BUCKET_KEY).toString();
String truststoreS3Path = appProperties.get(Helpers.TRUSTSTORE_S3_PATH_KEY).toString();
String keystorePassSecret =
    appProperties.get(Helpers.KEYSTORE_PASS_SECRET_KEY).toString();
String keystorePassSecretField =
    appProperties.get(Helpers.KEYSTORE_PASS_SECRET_FIELD_KEY).toString();

// region, etc..
builder.setProperty("config.providers.s3import.param.region", region);

// properties
builder.setProperty("ssl.truststore.location", "${s3import:" + region + ":" +
    truststoreS3Bucket + "/" + truststoreS3Path + "}");
builder.setProperty("ssl.keystore.type", "PKCS12");
builder.setProperty("ssl.keystore.location", "${s3import:" + region + ":" +
    keystoreS3Bucket + "/" + keystoreS3Path + "}");
builder.setProperty("ssl.keystore.password", "${secretsmanager:" + keystorePassSecret +
    ":" + keystorePassSecretField + "}");
builder.setProperty("ssl.key.password", "${secretsmanager:" + keystorePassSecret + ":" +
    keystorePassSecretField + "}");
...
```

[Puede encontrar más detalles y un tutorial aquí.](#)

APIs heredadas SourceFunction

Si utiliza las SourceFunction API heredadas, su aplicación utilizará esquemas de serialización y deserialización personalizados que anularán el open método para cargar el almacén de confianza

personalizado. Esto hace que el almacén de confianza esté disponible para la aplicación después de que la aplicación se reinicie o sustituya los subprocessos.

El almacén de confianza personalizado se recupera y almacena mediante el siguiente código:

```
public static void initializeKafkaTruststore() {
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
    URL inputUrl = classLoader.getResource("kafka.client.truststore.jks");
    File dest = new File("/tmp/kafka.client.truststore.jks");

    try {
        FileUtils.copyURLToFile(inputUrl, dest);
    } catch (Exception ex) {
        throw new FlinkRuntimeException("Failed to initialize Kafka truststore", ex);
    }
}
```

Note

Apache Flink requiere que el almacén de confianza esté en [formato JKS](#).


Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Cómo empezar \(DataStream API\)](#).

El siguiente tutorial muestra cómo conectarse de forma segura (cifrado en tránsito) a un clúster de Kafka que utiliza certificados de servidor emitidos por una autoridad de certificación (Certificate Authority, CA) personalizada, privada o incluso autohospedada.

Para conectar cualquier cliente de Kafka de forma segura a través de TLS a un clúster de Kafka, el cliente de Kafka (como la aplicación Flink del ejemplo) debe confiar en toda la cadena de confianza que presentan los certificados de servidor del clúster de Kafka (desde la CA emisora hasta la CA de nivel raíz). Como ejemplo de un Truststore personalizado, utilizaremos un clúster de Amazon MSK con la autenticación TLS mutua (MTLS) habilitada. Esto implica que los nodos del clúster de MSK utilizan certificados de servidor emitidos por una Autoridad de Certificación Privada de Certificate Manager (ACM Private CA) de AWS que es privada para su cuenta y región y, por lo tanto, no

confía en el almacén de confianza predeterminado de la máquina virtual Java (JVM) que ejecuta la aplicación Flink.

 Note

- Un almacén de claves se utiliza para almacenar la clave privada y los certificados de identidad que una aplicación debe presentar tanto al servidor como al cliente para su verificación.
- Un almacén de confianza se utiliza para almacenar los certificados de las autoridades certificadas (Certified Authorities, CA) que verifican el certificado presentado por el servidor en una conexión SSL.

También puede usar la técnica de este tutorial para las interacciones entre una aplicación de Managed Service para Apache Flink y otras fuentes de Apache Kafka, como:

- Un clúster de Apache Kafka personalizado alojado en AWS ([Amazon EC2](#) o [Amazon EKS](#))
- Un clúster de [Confluent Kafka](#) alojado en AWS
- Un clúster de Kafka en las instalaciones al que se accede a través de una VPN [AWS Direct Connect](#)

Este tutorial contiene las siguientes secciones:

- [Crear una VPC con un clúster de Amazon MSK](#)
- [Cree un almacén de confianza personalizado y aplíquelo a su clúster](#)
- [Crear el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear la aplicación](#)
- [Configurar la aplicación](#)
- [Ejecutar la aplicación](#)
- [Probar la aplicación](#)

Crear una VPC con un clúster de Amazon MSK

Para crear una muestra de VPC y un clúster de Amazon MSK al que acceder desde la aplicación de Managed Service para Apache Flink, siga el tutorial [Introducción al uso de Amazon MSK](#).

Cuando complete el tutorial, también haga lo siguiente:

- En el [Paso 3: Crear un tema](#), repita el comando `kafka-topics.sh --create` para crear un tema de destino llamado `AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --
replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

Note

Si el `kafka-topics.sh` comando devuelve un `ZooKeeperClientTimeoutException`, compruebe que el grupo de seguridad del clúster de Kafka tenga una regla de entrada que permita todo el tráfico procedente de la dirección IP privada de la instancia del cliente.

- Registre la lista de servidores de arranque de su clúster. Puede obtener la lista de servidores de arranque con el siguiente comando (sustitúyalo por `ClusterArn` el ARN de su clúster de MSK):

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
  "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Al seguir los pasos de este tutorial y los tutoriales de requisitos previos, asegúrese de usar la región de AWS seleccionada en el código, los comandos y las entradas de la consola.

Cree un almacén de confianza personalizado y aplíquelo a su clúster

En esta sección, debe crear una entidad de certificación (Certificate Authority, CA) personalizada, utilizarla para generar un almacén de confianza personalizado y aplicarla a su clúster de MSK.

Para crear y aplicar su almacén de confianza personalizado, siga el tutorial sobre [autenticación de clientes](#) de la Guía para desarrolladores de Amazon Managed Streaming para Apache Kafka.

Crear el código de la aplicación

En esta sección, descargará y compilará el archivo JAR de la aplicación.

El código de la aplicación Java de este ejemplo está disponible en [GitHub](#). Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. El código de la aplicación se encuentra en los `amazon-kinesis-data-analytics-java-examples/CustomKeystore`. Puede examinar el código para familiarizarse con la estructura del código de Managed Service para Apache Flink.
4. Utilice la herramienta Maven de línea de comandos o el entorno de desarrollo que prefiera para crear el archivo JAR. Para compilar el archivo JAR utilizando la herramienta Maven de línea de comandos, introduzca lo siguiente:

```
mvn package -Dflink.version=1.15.3
```

Si la compilación es correcta, se crea el siguiente archivo:

```
target/flink-app-1.0-SNAPSHOT.jar
```

Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Cargar el código de Java de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en el tutorial [Cómo empezar \(DataStream API\)](#).

Note

Si ha eliminado el bucket de Amazon S3 del tutorial de Introducción, vuelva a seguir el paso [the section called “Cargar el código de Java de streaming de Apache Flink”](#).

1. En la consola de Amazon S3, elija el <username>bucket ka-app-code- y, a continuación, seleccione Upload.
2. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `flink-app-1.0-SNAPSHOT.jar` que creó en el paso anterior.
3. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink versión 1.15.2.
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear un Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`

- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `flink-app-1.0-SNAPSHOT.jar`.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM `kinesis-analytics-MyApplication-us-west-2`.

Note

Al especificar los recursos de la aplicación mediante la consola (como registros o una VPC), la consola modifica la función de ejecución de la aplicación para conceder permiso de acceso a esos recursos.

4. En Propiedades, elija Añadir grupo. Introduzca las siguientes propiedades:

ID de grupo	Clave	Valor
KafkaSource	tema	AWSKafkaTutorialTopic
KafkaSource	bootstrap.servers	<i>La lista de servidores de arranque que guardó anteriormente</i>
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.contraseña	changeit

Note

El `ssl.truststore.password` del certificado predeterminado es "changeit"; no necesita cambiar este valor si utiliza el certificado predeterminado.

Vuelva a seleccionar Añadir grupo. Introduzca las siguientes propiedades:

ID de grupo	Clave	Valor
KafkaSink	tema	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	<i>La lista de servidores de arranque que guardó anteriormente</i>
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.contraseña	changeit
KafkaSink	transacción.timeout.ms	1 000

El código de la aplicación lee las propiedades de la aplicación anteriores para configurar el origen y el receptor que se utilizan para interactuar con la VPC y el clúster de Amazon MSK. Para obtener más información acerca de las propiedades, consulte [Propiedades de tiempo de ejecución](#).

5. En Instantáneas, seleccione Desactivar. Esto facilitará la actualización de la aplicación sin cargar datos de estado de la aplicación no válidos.
6. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
7. Para el CloudWatch registro, active la casilla Activar.

8. En la sección Nube privada virtual (VPC), elija la VPC que desea asociar a la aplicación. Elija las subredes y el grupo de seguridad asociados a la VPC que desee que utilice la aplicación para acceder a los recursos de la VPC.
9. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación.

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Probar la aplicación

En esta sección, se escriben los registros en el tema de origen. La aplicación lee los registros del tema de origen y los escribe en el tema de destino. Para comprobar que la aplicación funciona, escriba los registros en el tema de origen y lea los registros del tema de destino.

Para escribir y leer registros de los temas, siga los pasos del [Paso 6: Producir y consumir datos](#) del tutorial [Cómo empezar a utilizar Amazon MSK](#).

Para leer el tema de destino, utilice el nombre del tema de destino en lugar del tema de origen en la segunda conexión al clúster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --  
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-  
beginning
```


Si no aparece ningún registro en el tema de destino, consulte la sección [No puede obtener acceso a los recursos de una VPC](#) del tema [Solución de problemas](#).

Ejemplos de Python

En los siguientes ejemplos se muestra cómo crear aplicaciones usando Python mediante la API de Tabla de Apache Flink.

Temas

- [Ejemplo: creación de una ventana de saltos de tamaño constante en Python](#)
- [Ejemplo: creación de una ventana deslizante en Python](#)
- [Ejemplo: enviar datos de streaming a Amazon S3 en Python](#)

Ejemplo: creación de una ventana de saltos de tamaño constante en Python

En este ejercicio, creará una aplicación de Python Managed Service para Apache Flink que agrega datos mediante una ventana de salto de tamaño constante.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Introducción \(Python\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Comprima y cargue el código de Python de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`)
- Un bucket de Amazon S3 para almacenar el código de la aplicación (`ka-app-code-<username>`)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a sus flujos de datos **ExampleInputStream** y **ExampleOutputStream**.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

Note

El script de Python en esta sección usa AWS CLI. Debe configurar sus AWS CLI para usar las credenciales de su cuenta y la región predeterminada. Para configurar su AWS CLI, escriba lo siguiente:

```
aws configure
```

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/python/TumblingWindow`.

El código de la aplicación se encuentra en el archivo `tumbling-windows.py`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de tabla de Kinesis para leer del flujo de origen. El siguiente fragmento llama a la función `create_table` para crear el origen de la tabla de Kinesis:

```
table_env.execute_sql(
    create_input_table(input_table_name, input_stream, input_region,
        stream_initpos)
)
```

La función `create_table` utiliza un comando SQL para crear una tabla respaldada por el origen de streaming:

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time TIMESTAMP(3),
        WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{1}',
        'aws.region' = '{2}',
        'scan.stream.initpos' = '{3}',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """ .format(table_name, stream_name, region, stream_initpos)
```

- La aplicación utiliza el operador Tumble para agregar registros dentro de una ventana de saltos de tamaño constante específica y devolver los registros agregados como un objeto de tabla:

```
tumbling_window_table = (  
    input_table.window(  
        Tumble.over("10.seconds").on("event_time").alias("ten_second_window")  
    )  
    .group_by("ticker, ten_second_window")  
    .select("ticker, price.min as price, to_string(ten_second_window.end) as  
    event_time")
```

- La aplicación utiliza el conector Kinesis Flink, del [flink-sql-connector-kinesis-1.15.2.jar](#).

Comprima y cargue el código de Python de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. Utilice la aplicación de compresión que prefiera para comprimir los archivos `tumbling-windows.py` y `flink-sql-connector-kinesis-1.15.2.jar`. Dé el nombre `myapp.zip` al archivo.
2. En la consola de Amazon S3, elija el `<username>bucket ka-app-code-` y, a continuación, seleccione Upload.
3. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `myapp.zip` que creó en el paso anterior.
4. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink


Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.


3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:

- En Nombre de la aplicación, escriba **MyApplication**.
- En Tiempo de ejecución, escriba Apache Flink.

 Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

 Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `myapp.zip`.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **`kinesis-analytics-MyApplication-us-west-2`**.

4. En Propiedades, elija Añadir grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

Seleccione Guardar.

6. En Propiedades, elija Añadir grupo nuevamente.
7. Introduzca lo siguiente:

ID de grupo	Clave	Valor
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. En Propiedades, elija Añadir grupo nuevamente. En Nombre de grupo, introduzca **kinesis.analytics.flink.run.options**. Este grupo de propiedades especiales le indica a su aplicación dónde encontrar sus recursos de código. Para obtener más información, consulte [Especificar sus archivos de código](#).
9. Introduzca lo siguiente:

ID de grupo	Clave	Valor
kinesis.analytics.flink.run.options	python	tumbling-windows.py

ID de grupo	Clave	Valor
kinesis.analytics.flink.run.options	jarfile	flink-sql-connector-kinesis-1.15.2.jar

10. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
11. Para el CloudWatch registro, active la casilla Activar.
12. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: /aws/kinesis-analytics/MyApplication
- Flujo de registro: kinesis-analytics-log-stream

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (**012345678901**) por el ID de su cuenta.

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



```

"Statement": [
  {
    "Sid": "ReadCode",
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "logs:DescribeLogGroups",
      "s3:GetObjectVersion"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*",
      "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
    ]
  },
  {
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": "logs:DescribeLogStreams",
    "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": "logs:PutLogEvents",
    "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
  },
  {
    "Sid": "ListCloudwatchLogGroups",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  }
]

```

```
    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
  ]
}
```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Puede comprobar las métricas del servicio gestionado para Apache Flink en la CloudWatch consola para comprobar que la aplicación funciona.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana de salto de tamaño constante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija. MyApplication
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: creación de una ventana deslizante en Python

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Introducción \(Python\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Comprima y cargue el código de Python de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`)
- Un bucket de Amazon S3 para almacenar el código de la aplicación (`ka-app-code-<username>`)

Puede crear los flujos de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne un nombre a sus flujos de datos **ExampleInputStream** y **ExampleOutputStream**.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo **ka-app-code-*<username>***.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

Note

El script de Python en esta sección usa AWS CLI. Debe configurar sus AWS CLI para usar las credenciales de su cuenta y la región predeterminada. Para configurar su AWS CLI, escriba lo siguiente:

```
aws configure
```

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
```

```
kinesis_client.put_record(  
    StreamName=stream_name,  
    Data=json.dumps(data),  
    PartitionKey="partitionkey")  
  
if __name__ == '__main__':  
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/>amazon-kinesis-data-analytics-java-examples
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/python/SlidingWindow`.

El código de la aplicación se encuentra en el archivo `sliding-windows.py`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de tabla de Kinesis para leer del flujo de origen. El siguiente fragmento llama a la función `create_input_table` para crear el origen de la tabla de Kinesis:

```
table_env.execute_sql(  
    create_input_table(input_table_name, input_stream, input_region,  
    stream_initpos)
```

)

La función `create_input_table` utiliza un comando SQL para crear una tabla respaldada por el origen de streaming:

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time TIMESTAMP(3),
        WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{1}',
        'aws.region' = '{2}',
        'scan.stream.initpos' = '{3}',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """ .format(table_name, stream_name, region, stream_initpos)
}
```

- La aplicación utiliza el operador `Slide` para agregar registros dentro de una ventana deslizante específica y devolver los registros agregados como un objeto de tabla:

```
sliding_window_table = (
    input_table
    .window(
        Slide.over("10.seconds")
        .every("5.seconds")
        .on("event_time")
        .alias("ten_second_window")
    )
    .group_by("ticker, ten_second_window")
    .select("ticker, price.min as price, to_string(ten_second_window.end) as
event_time")
)
```

- [La aplicación utiliza el conector Kinesis Flink, del archivo -1.15.2.jar. flink-sql-connector-kinesis](#)

Comprima y cargue el código de Python de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

En esta sección, se describe cómo empaquetar la aplicación de Python.

1. Utilice la aplicación de compresión que prefiera para comprimir los archivos `sliding-windows.py` y `flink-sql-connector-kinesis-1.15.2.jar`. Dé el nombre `myapp.zip` al archivo.
2. En la consola de Amazon S3, elija el `<username>bucket ka-app-code-` y, a continuación, seleccione Upload.
3. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `myapp.zip` que creó en el paso anterior.
4. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la aplicación, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink.

Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `myapp.zip`.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM `kinesis-analytics-MyApplication-us-west-2`.
4. En Propiedades, elija Añadir grupo.
5. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
<code>consumer.config.0</code>	<code>input.stream.name</code>	<code>ExampleInputStream</code>
<code>consumer.config.0</code>	<code>aws.region</code>	<code>us-west-2</code>

ID de grupo	Clave	Valor
consumer.config.0	scan.stream.initpos	LATEST

Seleccione Guardar.

- En Propiedades, elija Añadir grupo nuevamente.
- Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

- En Propiedades, elija Añadir grupo nuevamente. En Nombre de grupo, introduzca **kinesis.analytics.flink.run.options**. Este grupo de propiedades especiales le indica a su aplicación dónde encontrar sus recursos de código. Para obtener más información, consulte [Especificar sus archivos de código](#).
- Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
kinesis.analytics.flink.run.options	python	sliding-windows.py
kinesis.analytics.flink.run.options	jarfile	flink-sql-connector-kinesis_1.15.2.jar

- En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
- Para el CloudWatch registro, active la casilla Activar.
- Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (**012345678901**) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
      ]
    }
  ]
}
```

```

    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": "logs:DescribeLogStreams",
      "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
  ]
}

```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Puede comprobar las métricas del servicio gestionado para Apache Flink en la CloudWatch consola para comprobar que la aplicación funciona.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana deslizante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija. MyApplication
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: enviar datos de streaming a Amazon S3 en Python

En este ejercicio, creará una aplicación de Python Managed Service para Apache Flink que transmita datos a un receptor de Amazon Simple Storage Service.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Introducción \(Python\)](#).

Este tema contiene las siguientes secciones:

- [Crear recursos dependientes](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargue y examine el código de la aplicación](#)
- [Comprima y cargue el código de Python de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Eliminación de recursos de AWS](#)

Crear recursos dependientes

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, debe crear los siguientes recursos dependientes:

- Un flujo de datos de Kinesis (`ExampleInputStream`)
- Un bucket de Amazon S3 para almacenar el código y los resultados de la aplicación (`ka-app-code-<username>`)

Note

Managed Service para Apache Flink no puede escribir datos en Amazon S3 con el cifrado del lado del servidor activado en Managed Service para Apache Flink.

Puede crear el flujo de Kinesis y el bucket de Amazon S3 usando la consola. Si desea obtener instrucciones para crear estos recursos, consulte los siguientes temas:

- [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams. Asigne el nombre `ExampleInputStream` a su flujo de datos.
- [¿Cómo se puede crear un bucket de S3?](#) en la guía de usuario de Amazon Simple Storage Service. Asigne al bucket de Amazon S3 un nombre único globalmente añadiendo su nombre de inicio de sesión, por ejemplo `ka-app-code-<username>`.

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

Note

El script de Python en esta sección usa AWS CLI. Debe configurar sus AWS CLI para usar las credenciales de su cuenta y la región predeterminada. Para configurar su AWS CLI, escriba lo siguiente:

```
aws configure
```

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
```



```
print(data)
kinesis_client.put_record(
    StreamName=stream_name,
    Data=json.dumps(data),
    PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Ejecute el script `stock.py`:

```
$ python stock.py
```

Mantenga el script en ejecución mientras completa el resto del tutorial.

Descargue y examine el código de la aplicación

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/python/S3Sink`.

El código de la aplicación se encuentra en el archivo `streaming-file-sink.py`. Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- La aplicación utiliza un origen de tabla de Kinesis para leer del flujo de origen. El siguiente fragmento llama a la función `create_source_table` para crear el origen de la tabla de Kinesis:

```
table_env.execute_sql(
    create_source_table(input_table_name, input_stream, input_region,
    stream_initpos)
)
```

La función `create_source_table` utiliza un comando SQL para crear una tabla respaldada por el origen de streaming

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

- La aplicación utiliza el conector `filesystem` para enviar registros a un bucket de Amazon S3:

```
def create_sink_table(table_name, bucket_name):
    return """ CREATE TABLE {0} (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time VARCHAR(64)
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector'='filesystem',
```

```
'path'='s3a://{1}/',
'format'='json',
'sink.partition-commit.policy.kind'='success-file',
'sink.partition-commit.delay' = '1 min'
) """ .format(table_name, bucket_name)
```

- [La aplicación utiliza el conector Kinesis Flink, del archivo -1.15.2.jar. flink-sql-connector-kinesis](#)

Comprima y cargue el código de Python de streaming de Apache Flink

En esta sección, cargará su código de aplicación en el bucket de Amazon S3 que creó en la sección [Crear recursos dependientes](#).

1. [Utilice la aplicación de compresión que prefiera para comprimir los archivos -1.15.2.jar y -1.15.2.jar. streaming-file-sink.py flink-sql-connector-kinesis](#) Dé el nombre `myapp.zip` al archivo.
2. En la consola de Amazon S3, elija el <username>bucket ka-app-code- y, a continuación, seleccione Upload.
3. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `myapp.zip` que creó en el paso anterior.
4. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la aplicación, escriba **MyApplication**.
 - En Tiempo de ejecución, escriba Apache Flink.

Note

Managed Service para Apache Flink utiliza la versión 1.15.2 de Apache Flink.

- Deje el menú desplegable de versión como Apache Flink versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `myapp.zip`.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM `kinesis-analytics-MyApplication-us-west-2`.
4. En Propiedades, elija Añadir grupo.
5. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

Seleccione Guardar.

- En Propiedades, elija Añadir grupo nuevamente. En Nombre de grupo, introduzca **kinesis.analytics.flink.run.options**. Este grupo de propiedades especiales le indica a su aplicación dónde encontrar sus recursos de código. Para obtener más información, consulte [Especificar sus archivos de código](#).
- Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
kinesis.analytics.flink.run.options	python	streaming-file-sink.py
kinesis.analytics.flink.run.options	jarfile	S3Sink/lib/flink-sql-connector-kinesis-1.15.2.jar

- En Propiedades, elija Añadir grupo nuevamente. En Nombre de grupo, introduzca **sink.config.0**. Este grupo de propiedades especiales le indica a su aplicación dónde encontrar sus recursos de código. Para obtener más información, consulte [Especificar sus archivos de código](#).
- Introduzca las siguientes propiedades y valores de la aplicación: (sustituya *bucket-name* por el nombre real de su bucket de Amazon S3).

ID de grupo	Clave	Valor
sink.config.0	output.bucket.name	<i>bucket-name</i>

10. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
11. Para el CloudWatch registro, active la casilla Activar.
12. Seleccione Actualizar.

Note

Si decide habilitar el CloudWatch registro, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros automáticamente. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Este flujo de registro se utiliza para supervisar la aplicación. No es el mismo flujo de registro que utiliza la aplicación para enviar los resultados.

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (**012345678901**) por el ID de su cuenta.

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

```

        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
    ]
},
{
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": "logs:DescribeLogStreams",
    "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
},
{
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": "logs:PutLogEvents",
    "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
},
{
    "Sid": "ListCloudwatchLogGroups",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
},
{
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
},
{
    "Sid": "WriteObjects",
    "Effect": "Allow",
    "Action": [
        "s3:Abort*",

```

```
        "s3:DeleteObject*",
        "s3:GetObject*",
        "s3:GetBucket*",
        "s3:List*",
        "s3:ListBucket",
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:s3:::ka-app-code-<username>",
        "arn:aws:s3:::ka-app-code-<username>/*"
    ]
}
]
```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Puede comprobar las métricas del servicio gestionado para Apache Flink en la CloudWatch consola para comprobar que la aplicación funciona.

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana deslizante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar su flujo de datos de Kinesis](#)
- [Eliminar objetos y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>

2. en el panel Servicio gestionado para Apache Flink, elija. MyApplication
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar su flujo de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStream página, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.

Eliminar objetos y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo de Scala

En los siguientes ejemplos se muestra cómo crear aplicaciones usando Scala con Apache Flink.

Temas

- [Ejemplo: creación de una ventana de saltos de tamaño constante en Scala](#)
- [Ejemplo: creación de una ventana deslizante en Scala](#)
- [Ejemplo: enviar datos de streaming a Amazon S3 en Scala](#)

Ejemplo: creación de una ventana de saltos de tamaño constante en Scala

Note

A partir de la versión 1.15, Flink es gratuito para Scala. Las aplicaciones ahora pueden usar la API de Java desde cualquier versión de Scala. Flink sigue utilizando Scala internamente en algunos componentes clave, pero no lo expone al cargador de clases del código de usuario. Por eso, los usuarios deben agregar las dependencias de Scala a sus archivos jar. Para obtener más información sobre los cambios de Scala en Flink 1.15, consulte [Scala Free in One Fifteen](#).

En este ejercicio, creará una aplicación de streaming sencilla que utiliza Scala 3.2.0 y la API Java de Flink. La aplicación lee los datos de la transmisión de Kinesis, los agrega mediante ventanas deslizantes y escribe los resultados para generar la transmisión de Kinesis.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Introducción \(Scala\)](#).

Este tema contiene las siguientes secciones:

- [Descargue y examine el código de la aplicación](#)
- [Compilar y cargar el código de la aplicación](#)
- [Crear y ejecutar la aplicación \(consola\)](#)
- [Compilar y ejecutar la aplicación \(CLI\)](#)

- [Actualizar el código de la aplicación](#)
- [Eliminación de recursos de AWS](#)

Descargue y examine el código de la aplicación

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo `build.sbt` contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.scala` contiene el método principal que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {  
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")  
  
  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,  
    defaultInputStreamName),  
    new SimpleStringSchema, inputProperties)  
}
```

La aplicación también utiliza un receptor de Kinesis para escribir en el flujo de resultado. El siguiente fragmento crea el receptor de Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- La aplicación utiliza el operador ventana para encontrar el recuento de valores de cada símbolo de cotización en una ventana de salto de tamaño constante de 5 segundos. El siguiente código crea el operador y envía los datos agregados a un nuevo receptor de flujo de datos de Kinesis:

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Int](jsonNode.get("ticker").toString, 1)
  }
  .returns(Types.TUPLE(Types.STRING, Types.INT))
  .keyBy(v => v.f0) // Logically partition the stream for each ticker
  .window(TumblingProcessingTimeWindows.of(Time.seconds(10)))
  .sum(1) // Sum the number of tickers per partition
  .map { value => value.f0 + "," + value.f1.toString + "\n" }
  .sinkTo(createSink)
```

- La aplicación crea conectores de origen y receptor para acceder a recursos externos mediante un `StreamExecutionEnvironment` objeto.
- La aplicación crea conectores de origen y recepción mediante las propiedades dinámicas de la aplicación. Las propiedades de tiempo de ejecución de la aplicación se leen para configurar los conectores. Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar y cargar el código de la aplicación

En esta sección, compilará y cargará el código de su aplicación a un bucket de Amazon S3.

Compilar el código de la aplicación

Utilice la herramienta de creación [SBT](#) para crear el código Scala para la aplicación. Para instalar SBT, consulte [Instalar sbt con configuración de cs](#). También deberá instalar el Kit de desarrollo de Java (JDK). Consulte los [Requisitos previos para completar los ejercicios](#).

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar su código con SBT:

```
sbt assembly
```

2. Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

Cargar el código de Scala de streaming de Apache Flink

En esta sección, creará un bucket de Amazon S3 y cargará el código de la aplicación.

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket
3. Escriba `ka-app-code- \langle username \rangle` en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. Abra el bucket `ka-app-code- \langle username \rangle` y elija Cargar.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `tumbling-window-scala-1.0.jar` que creó en el paso anterior.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My Scala test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

Utilice el siguiente procedimiento para configurar la aplicación.

Para configurar la aplicación

1. En la MyApplication página, elija Configurar.

2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **tumbling-window-scala-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, elija Añadir grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

Seleccione Guardar.

6. En Propiedades, elija Añadir grupo nuevamente.
7. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

8. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
9. Para el CloudWatch registro, active la casilla Activar.
10. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Editar la política de IAM

Edite la política de IAM para añadir los permisos para acceder al bucket de Amazon S3.

Para editar la política de IAM para añadir los permisos para el bucket de S3

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/tumbling-window-scala-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
```



```

    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
  },
  {
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]

```

```
}
```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

Para detener la aplicación, en la MyApplication página, seleccione Detener. Confirme la acción.

Compilar y ejecutar la aplicación (CLI)

En esta sección, se utiliza AWS Command Line Interface para crear y ejecutar la aplicación Managed Service para Apache Flink. Utilice el comando AWS CLI `kinesisanalyticsv2` para crear aplicaciones de Managed Service para Apache Flink e interactuar con ellas.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción de lectura en el flujo de origen y otra que concede permisos para las acciones de escritura en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

`AKReadStreamWriteSinkStream`. Reemplace **username** por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Reemplace el ID de la cuenta en el nombre de recurso de Amazon (ARN) (**012345678901**) por el ID de su cuenta. El rol de ejecución del servicio **MF-stream-rw-role** debe adaptarse a la función específica del cliente.

```
{
```

```

"ApplicationName": "tumbling_window",
"ApplicationDescription": "Scala tumbling window application",
"RuntimeEnvironment": "FLINK-1_15",
"ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
"ApplicationConfiguration": {
  "ApplicationCodeConfiguration": {
    "CodeContent": {
      "S3ContentLocation": {
        "BucketARN": "arn:aws:s3:::ka-app-code-username",
        "FileKey": "tumbling-window-scala-1.0.jar"
      }
    },
    "CodeContentType": "ZIPFILE"
  },
  "EnvironmentProperties": {
    "PropertyGroups": [
      {
        "PropertyGroupId": "ConsumerConfigProperties",
        "PropertyMap" : {
          "aws.region" : "us-west-2",
          "stream.name" : "ExampleInputStream",
          "flink.stream.initpos" : "LATEST"
        }
      },
      {
        "PropertyGroupId": "ProducerConfigProperties",
        "PropertyMap" : {
          "aws.region" : "us-west-2",
          "stream.name" : "ExampleOutputStream"
        }
      }
    ]
  },
  "CloudWatchLoggingOptions": [
    {
      "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
    }
  ]
}

```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial: Cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, seleccione Roles y, a continuación, seleccione Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS
4. En Elegir el servicio que usará este rol, elija Kinesis.
5. En Seleccione su caso de uso, elija Managed Service para Apache Flink.
6. Elija Siguiente: permisos.
7. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
8. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos para el rol

9. Asocie una política de permisos al rol.

Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de

datos de Kinesis. Para asociar la política que ha creado en el paso anterior: [Crear una política de permisos](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la política **AKReadSourceStreamWriteSinkStream** y, a continuación, elija Asociar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación

Guardé el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (nombre de usuario) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de muestra (012345678901) en el rol de ejecución del servicio por el ID de su cuenta. El `ServiceExecutionRole` debe incluir el rol de usuario de IAM que creó en la sección anterior.

```
"ApplicationName": "tumbling_window",
  "ApplicationDescription": "Scala getting started application",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "tumbling-window-scala-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    }
  },
```

```

"EnvironmentProperties": {
  "PropertyGroups": [
    {
      "PropertyGroupId": "ConsumerConfigProperties",
      "PropertyMap" : {
        "aws.region" : "us-west-2",
        "stream.name" : "ExampleInputStream",
        "flink.stream.initpos" : "LATEST"
      }
    },
    {
      "PropertyGroupId": "ProducerConfigProperties",
      "PropertyMap" : {
        "aws.region" : "us-west-2",
        "stream.name" : "ExampleOutputStream"
      }
    }
  ]
},
"CloudWatchLoggingOptions": [
  {
    "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
  }
]
}

```

[CreateApplication](#) Ejecútelo con la siguiente solicitud para crear la aplicación:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{
```

```
"ApplicationName": "tumbling_window",
"RunConfiguration": {
  "ApplicationRestoreConfiguration": {
    "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
  }
}
}
```

2. Ejecute la acción `StartApplication` con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "tumbling_window"
}
```

2. Ejecute la acción `StopApplication` con la solicitud anterior para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso de CloudWatch registros con su aplicación, consulte [Configuración del registro de aplicaciones](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{
  "ApplicationName": "tumbling_window",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleInputStream",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleOutputStream"
          }
        }
      ]
    }
  }
}
```

2. Ejecute la acción `UpdateApplication` con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```


Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de códigos, utilice la acción [UpdateApplicationCLI](#).

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [Crear recursos dependientes](#).

```
{
  "ApplicationName": "tumbling_window",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "tumbling-window-scala-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"
        }
      }
    }
  }
}
```

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana de salto de tamaño constante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: creación de una ventana deslizante en Scala

Note

A partir de la versión 1.15, Flink es gratuito para Scala. Las aplicaciones ahora pueden usar la API de Java desde cualquier versión de Scala. Flink sigue utilizando Scala internamente en algunos componentes clave, pero no lo expone al cargador de clases del código de usuario. Por eso, los usuarios deben agregar las dependencias de Scala a sus archivos jar. Para obtener más información sobre los cambios de Scala en Flink 1.15, consulte [Scala Free in One Fifteen](#).

En este ejercicio, creará una aplicación de streaming sencilla que utiliza Scala 3.2.0 y la API Java de Flink. DataStream La aplicación lee los datos de la transmisión de Kinesis, los agrega mediante ventanas deslizantes y escribe los resultados para generar la transmisión de Kinesis.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Introducción \(Scala\)](#).

Este tema contiene las siguientes secciones:

- [Descargue y examine el código de la aplicación](#)
- [Compilar y cargar el código de la aplicación](#)
- [Crear y ejecutar la aplicación \(consola\)](#)
- [Compilar y ejecutar la aplicación \(CLI\)](#)
- [Actualizar el código de la aplicación](#)
- [Eliminación de recursos de AWS](#)

Descargue y examine el código de la aplicación

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo `build.sbt` contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.scala` contiene el método principal que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
    defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

La aplicación también utiliza un receptor de Kinesis para escribir en el flujo de resultado. El siguiente fragmento crea el receptor de Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
    defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- La aplicación utiliza el operador ventana para encontrar el conteo de valores para cada símbolo de cotización en una ventana de 10 segundos que se desliza 5 segundos. El siguiente código crea el operador y envía los datos agregados a un nuevo receptor de flujo de datos de Kinesis:

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Double](jsonNode.get("ticker").toString,
    jsonNode.get("price").asDouble)
  }
  .returns(Types.TUPLE(Types.STRING, Types.DOUBLE))
  .keyBy(v => v.f0) // Logically partition the stream for each word
  .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5)))
  .min(1) // Calculate minimum price per ticker over the window
  .map { value => value.f0 + String.format(":%.2f", value.f1) + "\n" }
  .sinkTo(createSink)
```

- La aplicación crea conectores de origen y receptor para acceder a recursos externos mediante un `StreamExecutionEnvironment` objeto.
- La aplicación crea conectores de origen y recepción mediante las propiedades dinámicas de la aplicación. Las propiedades de tiempo de ejecución de la aplicación se leen para configurar los conectores. Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar y cargar el código de la aplicación

En esta sección, compilará y cargará el código de su aplicación a un bucket de Amazon S3.

Compilar el código de la aplicación

Utilice la herramienta de creación [SBT](#) para crear el código Scala para la aplicación. Para instalar SBT, consulte [Instalar sbt con configuración de cs](#). También deberá instalar el Kit de desarrollo de Java (JDK). Consulte los [Requisitos previos para completar los ejercicios](#).

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar su código con SBT:

```
sbt assembly
```

2. Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

Cargar el código de Scala de streaming de Apache Flink

En esta sección, creará un bucket de Amazon S3 y cargará el código de la aplicación.

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket
3. Escriba `ka-app-code-<username>` en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.

6. Elija Crear bucket.
7. Abra el bucket `ka-app-code-<username>` y elija Cargar.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `sliding-window-scala-1.0.jar` que creó en el paso anterior.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My Scala test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

Utilice el siguiente procedimiento para configurar la aplicación.

Para configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `sliding-window-scala-1.0.jar..`
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM `kinesis-analytics-MyApplication-us-west-2`.
4. En Propiedades, elija Añadir grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
<code>ConsumerConfigProperties</code>	<code>input.stream.name</code>	<code>ExampleInputStream</code>
<code>ConsumerConfigProperties</code>	<code>aws.region</code>	<code>us-west-2</code>
<code>ConsumerConfigProperties</code>	<code>flink.stream.initpos</code>	<code>LATEST</code>

Seleccione Guardar.

6. En Propiedades, elija Añadir grupo nuevamente.
7. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

- En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
- Para el CloudWatch registro, active la casilla Activar.
- Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Editar la política de IAM

Edite la política de IAM para añadir los permisos para acceder al bucket de Amazon S3.

Para editar la política de IAM para añadir los permisos para el bucket de S3

- Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
- Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
- En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
- Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/sliding-window-scala-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [

```

```

        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

Para detener la aplicación, en la MyApplication página, seleccione Detener. Confirme la acción.

Compilar y ejecutar la aplicación (CLI)

En esta sección, se utiliza AWS Command Line Interface para crear y ejecutar la aplicación Managed Service para Apache Flink. Utilice el comando AWS CLI `kinesisanalyticsv2` para crear aplicaciones de Managed Service para Apache Flink e interactuar con ellas.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción de lectura en el flujo de origen y otra que concede permisos para las acciones de escritura en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

AKReadSourceStreamWriteSinkStream. Reemplace **username** por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Reemplace el ID de la cuenta en el nombre de recurso de Amazon (ARN) (**012345678901**) por el ID de su cuenta.

```
{
  "ApplicationName": "sliding_window",
  "ApplicationDescription": "Scala sliding window application",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "sliding-window-scala-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
```

```
        "aws.region" : "us-west-2",
        "stream.name" : "ExampleInputStream",
        "flink.stream.initpos" : "LATEST"
    }
},
{
    "PropertyGroupId": "ProducerConfigProperties",
    "PropertyMap" : {
        "aws.region" : "us-west-2",
        "stream.name" : "ExampleOutputStream"
    }
}
]
}
},
"CloudWatchLoggingOptions": [
    {
        "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
    }
]
}
```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial: Cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.


Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.

2. En el panel de navegación, seleccione Roles y, a continuación, seleccione Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS
4. En Elegir el servicio que usará este rol, elija Kinesis.
5. En Seleccione su caso de uso, elija Managed Service para Apache Flink.
6. Elija Siguiente: permisos.
7. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
8. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos para el rol

9. Asocie una política de permisos al rol.

 Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Para asociar la política que ha creado en el paso anterior: [Crear una política de permisos](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la política `AKReadSourceStreamWriteSinkStream` y, a continuación, elija Asociar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación

Guarde el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (nombre de usuario) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de muestra (012345678901) en el rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "sliding_window",
  "ApplicationDescription": "Scala sliding_window application",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "sliding-window-scala-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleInputStream",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleOutputStream"
          }
        }
      ]
    }
  },
  "CloudWatchLoggingOptions": [
```

```
{
  "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
}
]
```

[CreateApplication](#) Ejecútelo con la siguiente solicitud para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{
  "ApplicationName": "sliding_window",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

2. Ejecute la acción `StartApplication` con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "sliding_window"
}
```

2. Ejecute la acción `StopApplication` con la solicitud anterior para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso de CloudWatch registros con su aplicación, consulte [Configuración del registro de aplicaciones](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "sliding_window",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2",
            "stream.name" : "ExampleInputStream",
```

```
        "flink.stream.initpos" : "LATEST"
      }
    },
    {
      "PropertyGroupId": "ProducerConfigProperties",
      "PropertyMap" : {
        "aws.region" : "us-west-2",
        "stream.name" : "ExampleOutputStream"
      }
    }
  ]
}
```

2. Ejecute la acción `UpdateApplication` con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de códigos, utilice la acción [UpdateApplicationCLI](#).

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la

versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (<username>) con el sufijo que haya elegido en la sección [Crear recursos dependientes](#).

```
{
  "ApplicationName": "sliding_window",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvpDU"
        }
      }
    }
  }
}
```

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana deslizante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Ejemplo: enviar datos de streaming a Amazon S3 en Scala

Note

A partir de la versión 1.15, Flink es gratuito para Scala. Las aplicaciones ahora pueden usar la API de Java desde cualquier versión de Scala. Flink sigue utilizando Scala internamente en algunos componentes clave, pero no lo expone al cargador de clases del código de usuario. Por eso, los usuarios deben agregar las dependencias de Scala a sus archivos jar. Para obtener más información sobre los cambios de Scala en Flink 1.15, consulte [Scala Free in One Fifteen](#).

En este ejercicio, creará una aplicación de streaming sencilla que utiliza Scala 3.2.0 y la API Java de Flink. La aplicación lee los datos de la transmisión de Kinesis, los agrega mediante ventanas deslizantes y escribe los resultados en S3.

Note

Para configurar los requisitos previos necesarios para este ejercicio, primero complete el ejercicio [Introducción \(Scala\)](#). Solo necesita crear una carpeta adicional **data/** en el bucket de Amazon S3 ka-app-code-<username>.

Este tema contiene las siguientes secciones:

- [Descargue y examine el código de la aplicación](#)
- [Compilar y cargar el código de la aplicación](#)
- [Crear y ejecutar la aplicación \(consola\)](#)
- [Compilar y ejecutar la aplicación \(CLI\)](#)
- [Actualizar el código de la aplicación](#)
- [Eliminación de recursos de AWS](#)

Descargue y examine el código de la aplicación

El código de la aplicación Python para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Si aún no lo ha hecho, instale el cliente Git. Para obtener más información, consulte [Instalación de Git](#).
2. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/scala/S3Sink`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo `build.sbt` contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.scala` contiene el método principal que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
    defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

La aplicación también utiliza un `StreamingFileSink` para escribir en un bucket de Amazon S3:

```
def createSink: StreamingFileSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val s3SinkPath =
    applicationProperties.get("ProducerConfigProperties").getProperty("s3.sink.path")

  StreamingFileSink
    .forRowFormat(new Path(s3SinkPath), new SimpleStringEncoder[String]("UTF-8"))
    .build()
}
```

- La aplicación crea conectores de origen y receptor para acceder a recursos externos mediante un `StreamExecutionEnvironment` objeto.
- La aplicación crea conectores de origen y recepción mediante las propiedades dinámicas de la aplicación. Las propiedades de tiempo de ejecución de la aplicación se leen para configurar los conectores. Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar y cargar el código de la aplicación

En esta sección, compilará y cargará el código de su aplicación a un bucket de Amazon S3.

Compilar el código de la aplicación

Utilice la herramienta de creación [SBT](#) para crear el código Scala para la aplicación. Para instalar SBT, consulte [Instalar sbt con configuración de cs](#). También deberá instalar el Kit de desarrollo de Java (JDK). Consulte los [Requisitos previos para completar los ejercicios](#).

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar su código con SBT:

```
sbt assembly
```

2. Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

Cargar el código de Scala de streaming de Apache Flink

En esta sección, creará un bucket de Amazon S3 y cargará el código de la aplicación.

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket
3. Escriba `ka-app-code-<username>` en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.

6. Elija Crear bucket.
7. Abra el bucket `ka-app-code-<username>` y elija Cargar.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `s3-sink-scala-1.0.jar` que creó en el paso anterior.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje la versión como Apache Flink, versión 1.15.2 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Configurar la aplicación

Utilice el siguiente procedimiento para configurar la aplicación.

Para configurar la aplicación

1. En la MyApplication página, elija Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca `ka-app-code-<username>`.
 - En Ruta al objeto de Amazon S3, introduzca `s3-sink-scala-1.0.jar`.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM `kinesis-analytics-MyApplication-us-west-2`.
4. En Propiedades, elija Añadir grupo.
5. Introduzca lo siguiente:

ID de grupo	Clave	Valor
<code>ConsumerConfigProperties</code>	<code>input.stream.name</code>	<code>ExampleInputStream</code>
<code>ConsumerConfigProperties</code>	<code>aws.region</code>	<code>us-west-2</code>
<code>ConsumerConfigProperties</code>	<code>flink.stream.initpos</code>	<code>LATEST</code>

Seleccione Guardar.

6. En Propiedades, elija Añadir grupo.
7. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ProducerConfigProperties	s3.sink.path	s3a://ka-app-code- <user-name> /data

8. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
9. Para el CloudWatch registro, active la casilla Activar.
10. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: /aws/kinesis-analytics/MyApplication
- Flujo de registro: kinesis-analytics-log-stream

Editar la política de IAM

Edite la política de IAM para añadir los permisos para acceder al bucket de Amazon S3.

Para editar la política de IAM para añadir los permisos para el bucket de S3

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (**012345678901**) por el ID de su cuenta.

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

```
{
  "Sid": "ReadCode",
  "Effect": "Allow",
  "Action": [
    "s3:Abort*",
    "s3:DeleteObject*",
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*",
    "s3:ListBucket",
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::ka-app-code-<username>",
    "arn:aws:s3:::ka-app-code-<username>/*"
  ]
},
{
  "Sid": "DescribeLogGroups",
  "Effect": "Allow",
  "Action": [
    "logs:DescribeLogGroups"
  ],
  "Resource": [
    "arn:aws:logs:us-west-2:012345678901:log-group:*"
  ]
},
{
  "Sid": "DescribeLogStreams",
  "Effect": "Allow",
  "Action": [
    "logs:DescribeLogStreams"
  ],
  "Resource": [
    "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
  ]
},
{
  "Sid": "PutLogEvents",
  "Effect": "Allow",
  "Action": [
    "logs:PutLogEvents"
  ],
}
```

```
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  }
]
```

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

Para detener la aplicación, en la MyApplication página, seleccione Detener. Confirme la acción.

Compilar y ejecutar la aplicación (CLI)

En esta sección, se utiliza AWS Command Line Interface para crear y ejecutar la aplicación Managed Service para Apache Flink. Utilice el comando AWS CLI `kinesisanalyticsv2` para crear aplicaciones de Managed Service para Apache Flink e interactuar con ellas.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción de lectura en el flujo de origen y otra que concede permisos para las acciones de escritura en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que

se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

AKReadStreamWriteSinkStream. Reemplace **username** por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Reemplace el ID de la cuenta en el nombre de recurso de Amazon (ARN) (**012345678901**) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
      ]
    }
  ]
}
```

```

    ]
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial: Cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de

permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.


Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, seleccione Roles y, a continuación, seleccione Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS
4. En Elegir el servicio que usará este rol, elija Kinesis.
5. En Seleccione su caso de uso, elija Managed Service para Apache Flink.
6. Elija Siguiente: permisos.
7. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
8. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos para el rol

9. Asocie una política de permisos al rol.

 Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Para asociar la política que ha creado en el paso anterior: [Crear una política de permisos](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la política `AKReadSourceStreamWriteSinkStream` y, a continuación, elija Asociar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación

Guardé el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (nombre de usuario) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de muestra (012345678901) en el rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "s3_sink",
  "ApplicationDescription": "Scala tumbling window application",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "s3-sink-scala-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-west-2",
            "stream.name": "ExampleInputStream",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "s3.sink.path": "s3a://ka-app-code-<username>/data"
          }
        }
      ]
    }
  }
}
```



```
    }
  ]
}
},
"CloudWatchLoggingOptions": [
  {
    "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
  }
]
}
```

[CreateApplication](#) Ejecútelo con la siguiente solicitud para crear la aplicación:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{
  "ApplicationName": "s3_sink",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

2. Ejecute la acción `StartApplication` con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "s3_sink"
}
```

2. Ejecute la acción `StopApplication` con la solicitud anterior para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso de CloudWatch registros con su aplicación, consulte [Configuración del registro de aplicaciones](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "s3_sink",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
```


En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (<username>) con el sufijo que haya elegido en la sección [Crear recursos dependientes](#).

```
{
  "ApplicationName": "s3_sink",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "s3-sink-scala-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

Eliminación de recursos de AWS

En esta sección se incluyen los procedimientos para limpiar los recursos de AWS creados en el tutorial Ventana de salto de tamaño constante.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. en el panel Servicio gestionado para Apache Flink, elija MyApplication.

3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.

4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Seguridad en Amazon Managed Service para Apache Flink

En AWS, la seguridad en la nube es la máxima prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y un centro de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y el usuario. El [modelo de responsabilidad compartida](#) la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los [programas de conformidad de AWS](#). Para obtener más información acerca de los programas de conformidad que se aplican a Managed Service para Apache Flink, consulte [Servicios en el ámbito del programa de conformidad de AWS](#).
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Managed Service para Apache Flink. En los siguientes temas, se muestra cómo configurar Managed Service para Apache Flink para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros servicios de Amazon que lo pueden ayudar a monitorear y proteger sus recursos de Managed Service para Apache Flink.

Temas

- [Protección de los datos en Amazon Managed Service para Apache Flink](#)
- [Identidad y gestión de acceso para Amazon Managed Service para Apache Flink](#)
- [Supervisión de Managed Service para Apache Flink](#)
- [Validación de la conformidad de Amazon Managed Service para Apache Flink](#)
- [Resiliencia en Amazon Managed Service para Apache Flink](#)
- [Seguridad de la infraestructura en Amazon Managed Service para Apache Flink](#)
- [Mejores prácticas de seguridad para el servicio gestionado de Managed Service para Apache Flink](#)

Protección de los datos en Amazon Managed Service para Apache Flink

Puede proteger sus datos mediante las herramientas que proporciona AWS. Managed Service para Apache Flink puede funcionar con servicios que admiten el cifrado de datos, como Kinesis Data Firehose y Amazon S3.

Cifrado de datos en un servicio gestionado para Apache Flink

Cifrado en reposo

Tenga en cuenta lo siguiente sobre el cifrado de datos en reposo con Managed Service para Apache Flink::

- Puede cifrar los datos de la transmisión de datos entrante de Kinesis mediante [StartStreamEncryption](#). Para obtener más información, consulte [¿Qué es el cifrado en el servidor para flujo de datos de Kinesis?](#)
- Los datos de salida se pueden cifrar en reposo con Kinesis Data Firehose, que permite almacenar los datos en un bucket de Amazon S3 cifrado. Puede especificar la clave de cifrado que el bucket de Amazon S3 va a utilizar. Para obtener más información, consulte [Protección de los datos mediante el cifrado del servidor con claves administradas por KMS \(SSE-KMS\)](#).
- El servicio gestionado para Apache Flink puede leer desde cualquier fuente de streaming y escribir en cualquier destino de streaming o base de datos. Asegúrese de que sus fuentes y destinos cifran todos los datos en tránsito y los datos en reposo.
- El código de la aplicación se cifra en reposo.
- El almacenamiento de aplicaciones duradero se cifra en reposo.
- El almacenamiento de aplicaciones en ejecución se cifra en reposo.

Cifrado en tránsito

Managed Service para Apache Flink cifra todos los datos en tránsito. El cifrado en tránsito está habilitado para todas las aplicaciones de Managed Service para Apache Flink y no se puede deshabilitar.

Managed Service para Apache Flink cifra los datos en tránsito en los siguientes escenarios:

- Datos en tránsito de flujo de datos de Kinesis a Managed Service para Apache Flink.

- Datos en tránsito entre componentes internos de Managed Service para Apache Flink.
- Datos en tránsito entre Managed Service para Apache Flink y Kinesis Data Firehose.

Administración de claves

El cifrado de datos en Managed Service para Apache Flink utiliza claves administradas por el servicio. No se admiten las claves administradas por el cliente.

Identidad y gestión de acceso para Amazon Managed Service para Apache Flink

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda a los administradores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de Managed Service para Apache Flink. IAM es un servicio de Servicio de AWS que se puede utilizar sin cargo adicional.

Temas

- [Público](#)
- [Autenticación con identidades](#)
- [Administración de acceso mediante políticas](#)
- [Cómo funciona Amazon Managed Service para Apache Flink con IAM](#)
- [Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink](#)
- [Solución de problemas de identidad y acceso de Amazon Managed Service para Apache Flink](#)
- [Prevención de la sustitución confusa entre servicios](#)

Público

La forma en que utilice AWS Identity and Access Management (IAM) difiere en función del trabajo que realice en Managed Service para Apache Flink.

Usuario de servicio: si utiliza el servicio de Managed Service para Apache Flink para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. A medida que

utilice más características de Managed Service para Apache Flink a fin de realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en Managed Service para Apache Flink, consulte [Solución de problemas de identidad y acceso de Amazon Managed Service para Apache Flink](#).

Administrador de servicio: si está a cargo de los recursos de Managed Service para Apache Flink en su empresa, probablemente tenga acceso completo a Managed Service para Apache Flink. Su trabajo consiste en determinar a qué características y recursos de Managed Service para Apache Flink para la investigación deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador 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 su empresa puede utilizar IAM con Managed Service para Apache Flink, consulte [Cómo funciona Amazon Managed Service para Apache Flink con IAM](#).

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a Managed Service para Apache Flink. Para consultar ejemplos de políticas basadas en la identidad de Managed Service para Apache Flink que puede utilizar en IAM, consulte [Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink](#).

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado (haber iniciado sesión en AWS) como Usuario raíz de la cuenta de AWS, como un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión en AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad de AWS IAM Identity Center. Los usuarios (del IAM Identity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accede a AWS mediante la federación, está asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en la AWS Management Console o en el portal de acceso a AWS. Para obtener más información sobre el inicio de sesión en AWS, consulte [Cómo iniciar sesión en su Cuenta de AWS](#) en la Guía del usuario de AWS Sign-In.

Si accede a AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de la línea de comandos (CLI) para firmar criptográficamente las solicitudes mediante el uso de las credenciales. Si no utiliza las herramientas de AWS, debe firmar usted mismo las solicitudes. Para obtener más información sobre el método recomendado para la firma de solicitudes, consulte [Firma de solicitudes API de AWS](#) en la Guía del usuario de IAM.

Independientemente del método de autenticación que utilice, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, AWS le recomienda el uso de 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 [Uso de la autenticación multifactor \(MFA\) en AWS](#) en la Guía del usuario de IAM.

Usuario raíz de Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión que tiene acceso completo a todos los recursos y Servicios de AWS de la cuenta. Esta identidad recibe el nombre de usuario raíz de la Cuenta de AWS y se accede a ella iniciando sesión con el email y la contraseña que utilizó para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte [Tareas que requieren credenciales de usuario raíz](#) en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, solicite que los usuarios humanos, incluidos los que requieren acceso de administrador, utilicen la federación con un proveedor de identidades para acceder a Servicios de AWS utilizando credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidad web, el AWS Directory Service, el directorio del Identity Center, o cualquier usuario que acceda a Servicios de AWS utilizando credenciales proporcionadas a través de una fuente de identidad. Cuando identidades federadas acceden a las Cuentas de AWS, asumen roles y los roles proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el IAM Identity Center o puede conectarse y sincronizar con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus

aplicaciones y Cuentas de AWS. Para obtener más información, consulte [¿Qué es el IAM Identity Center?](#) en la Guía del usuario de AWS IAM Identity Center.

Usuarios y grupos de IAM

Un [usuario de IAM](#) es una identidad en su Cuenta de AWS que dispone de 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, consulte [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 puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo 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 permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte [Cuándo crear un usuario de IAM \(en lugar de un rol\)](#) en la Guía del Usuario de IAM.

Roles de IAM

Un [rol de IAM](#) es una identidad de tu Cuenta de AWS que dispone de permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente un rol de IAM en la AWS Management Console [cambiando de roles](#). Puede asumir un rol llamando a una operación de AWS CLI o de la API de AWS, o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte [Uso de roles de IAM](#) 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, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para

federación, consulte [Creación de un rol para un proveedor de identidades de terceros](#) en la Guía del Usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. El IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte [Conjuntos de permisos](#) en la Guía del usuario de AWS IAM Identity Center.

- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede 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. No obstante, con algunos Servicios de AWS se puede asociar una política directamente a un recurso (en lugar de utilizar un rol como representante). Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte [Cómo los roles de IAM difieren de las políticas basadas en recursos](#) en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan características de otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común 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 a servicios.
- Reenviar sesiones de acceso (FAS): cuando utiliza un rol o un usuario de IAM para llevar a cabo acciones en AWS, se le considera una entidad principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o 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, consulte [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.
- Rol vinculado al servicio: un rol vinculado al servicio es un tipo de rol de servicio que está vinculado a un Servicio de AWS. El servicio puede asumir el rol para realizar una acción en su

nombre. Los roles vinculados a servicios aparecen en la Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede utilizar un rol de IAM que le permita administrar credenciales temporales para las aplicaciones que se ejecutan en una instancia de EC2 y realizan solicitudes a la AWS CLI o a la API de AWS. Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia EC2. Para asignar un rol de AWS a una instancia de EC2 y ponerla a disposición de todas las aplicaciones, cree un perfil de instancia asociado a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia EC2 obtener credenciales temporales. Para obtener más información, consulte [Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias de Amazon EC2](#) en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte [Cuándo crear un rol de IAM \(en lugar de un usuario\)](#) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política es un objeto de AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando una entidad principal (sesión de rol, usuario o usuario raíz) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte [Información general de las políticas JSON](#) en la Guía del Usuario de IAM.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder permiso a los usuarios para realizar acciones en los recursos que necesiten, un administrador puede crear políticas de IAM. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con dicha política puede obtener información del rol de la AWS Management Console, la AWS CLI o la API de AWS.

Políticas basadas en identidades

Las políticas basadas en identidades son documentos de políticas de permisos JSON que puede adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. 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 asociar a varios usuarios, grupos y roles de su Cuenta de AWS. Las políticas administradas incluyen las políticas administradas de AWS y las 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. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#) en una política en función de recursos. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas de AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulte [Información general de Lista de control de acceso \(ACL\)](#) en la Guía para Desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite otros tipos de políticas adicionales menos frecuentes. 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 puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte [Límites de permisos para las entidades de IAM](#) en la Guía del Usuario de IAM.
- **Políticas de control de servicio (SCP):** las SCP son políticas de JSON que especifican los permisos máximos de una organización o una unidad organizativa en AWS Organizations. AWS Organizations es un servicio que le permite agrupar y administrar de manera centralizada varias Cuentas de AWS que posea su empresa. Si habilita todas las características en una empresa, entonces podrá aplicar políticas de control de servicio (SCP) a una o todas sus cuentas. Una SCP limita los permisos para las entidades de las cuentas de miembros, incluido cada Usuario raíz de la cuenta de AWS. Para obtener más información acerca de Organizations y las SCP, consulte [Funcionamiento de las SCP](#) en la Guía del usuario AWS Organizations.
- **Políticas de sesión:** las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte [Políticas de sesión](#) en la Guía del Usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información sobre cómo AWS decide si permite o no una solicitud cuando hay varios tipos de políticas implicados, consulte [Lógica de evaluación de políticas](#) en la Guía del usuario de IAM.

Cómo funciona Amazon Managed Service para Apache Flink con IAM

Antes de utilizar IAM para administrar el acceso a Managed Service para Apache Flink, conozca qué características de IAM se pueden utilizar con Managed Service para Apache Flink.

Características de IAM que puede utilizar con Amazon Managed Service para Apache Flink

Característica de IAM	Soporte de Managed Service para Apache Flink
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	No
ACL	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	No
Roles vinculados al servicio	No

Para obtener información general sobre cómo funcionan Managed Service para Apache Flink y otros servicios de AWS con la mayoría de las características de IAM, consulte [Servicios de AWS que funcionan con IAM](#) en la Guía del usuario de IAM.

Políticas basadas en identidad para los servicios gestionados de Amazon Flink

Compatibilidad con las políticas basadas en identidades	Sí
---	----

Las políticas basadas en identidades son documentos de políticas de permisos JSON que puede adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. 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 adjunto. Para obtener más información sobre los elementos que puede utilizar en una política JSON, consulte [Referencia de los elementos de las políticas JSON de IAM](#) en la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades de Managed Service para Apache Flink

Para consultar ejemplos de políticas basadas en la identidad de Managed Service para Apache Flink, consulte [Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink](#).

Políticas basadas en recursos dentro de Managed Service para Apache Flink

Compatibilidad con las políticas basadas en recursos	Sí
--	----

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#)

en una política en función de recursos. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o servicios de 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 la entidad principal y el recurso se encuentran en Cuentas de AWS diferentes, un 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 en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para más información, consulte [Cómo los roles de IAM difieren de las políticas basadas en recursos](#) en la Guía del Usuario de IAM.

Acciones de política para Managed Service para Apache Flink

Admite acciones de políticas

Sí

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para permitir o denegar el acceso en una política. Las acciones de la política generalmente tienen el mismo nombre que la operación de API de AWS 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 de las acciones de Managed Service para Apache Flink, consulte [Acciones definidas por Amazon Managed Service para Apache Flink](#) en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Managed Service para Apache Flink utilizan el siguiente prefijo antes de la acción:

```
Kinesis Analytics
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [  
  "Kinesis Analytics:action1",  
  "Kinesis Analytics:action2"  
]
```

Puede utilizar caracteres comodín (*) para especificar varias acciones. Por ejemplo, para especificar todas las acciones que comiencen con la palabra Describe, incluya la siguiente acción:

```
"Action": "Kinesis Analytics:Describe*"
```

Para consultar ejemplos de políticas basadas en la identidad de Managed Service para Apache Flink, consulte [Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink](#).

Recursos de políticas para Managed Service para Apache Flink

Admite recursos de políticas

Sí

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento Resource de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento Resource o NotResource. Como práctica recomendada, especifique un recurso utilizando el [Nombre de recurso de Amazon \(ARN\)](#). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de recursos de Managed Service para Apache Flink y sus ARN, consulte [Recursos definidos por Amazon Managed Service para Apache Flink](#) en la Referencia de autorizaciones de servicio. Para obtener información acerca de las acciones con las que puede especificar el ARN de cada recurso, consulte [Acciones definidas por Amazon Managed Service para Apache Flink](#).

Para consultar ejemplos de políticas basadas en la identidad de Managed Service para Apache Flink, consulte [Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink](#).

Claves de condición de política para Managed Service para Apache Flink

Admite claves de condición de políticas específicas del servicio	Sí
--	----

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen [operadores de condición](#), tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación lógica AND. Si especifica varios valores para una única clave de condición, AWS evalúa la condición con una operación OR lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder 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, consulte [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 globales de AWS, consulte [Claves de contexto de condición globales de AWS](#) en la Guía del Usuario de IAM.

Para obtener una lista de las claves de condición de Managed Service para Apache Flink, consulte [Claves de condición de Amazon Managed Service para Apache Flink](#) en la Referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte [acciones definidas por Amazon Managed Service para Apache Flink](#).

Para consultar ejemplos de políticas basadas en la identidad de Managed Service para Apache Flink, consulte [Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink](#).

Listas de control de acceso (ACL) en Managed Service para Apache Flink

Admite las ACL

No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Control de acceso basado en atributos (ABAC) con Managed Service para Apache Flink

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 entidades de IAM (usuarios o roles) y a muchos recursos de AWS. 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 [¿Qué es ABAC?](#) en la Guía del Usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte [Uso del control de acceso basado en atributos \(ABAC\)](#) en la Guía del Usuario de IAM.

Uso de credenciales temporales con Managed Service para Apache Flink

Admite el uso de credenciales temporales	Sí
--	----

Algunos Servicios de AWS no funcionan cuando inicia sesión con credenciales temporales. Para obtener información adicional, incluida la información sobre qué Servicios de AWS funcionan con credenciales temporales, consulte [Servicios de AWS que funcionan con IAM](#) en la Guía del usuario de IAM.

Utilice credenciales temporales si inicia sesión en la AWS Management Console con cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accede a AWS utilizando el enlace de inicio de sesión único (SSO) de la empresa, ese proceso crea automáticamente credenciales temporales. También crea automáticamente credenciales temporales cuando inicia sesión en la consola como usuario y luego cambia de rol. Para obtener más información sobre el cambio de roles, consulte [Cambio a un rol \(consola\)](#) en la Guía del usuario de IAM.

Puede crear credenciales temporales de forma manual mediante la AWS CLI o la API de AWS. A continuación, puede usar esas credenciales temporales para acceder a AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de usar claves de acceso a largo plazo. Para más información, consulte [Credenciales de seguridad temporales en IAM](#).

Permisos de entidades principales entre servicios de Managed Service para Apache Flink

Admite Forward access sessions (FAS)	Sí
--------------------------------------	----

Cuando utiliza un usuario o un rol de IAM para llevar a cabo acciones en AWS, se lo considera una entidad principal. Cuando utiliza algunos servicios, es posible que realice una acción que

desencadene otra acción en un servicio diferente. FAS utiliza los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o 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, consulte [Reenviar sesiones de acceso](#).

Roles de servicio para Managed Service para Apache Flink

Compatible con roles de servicio	Sí
----------------------------------	----

Un rol de servicio es un [rol de IAM](#) que asume 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.

Warning

Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Managed Service para Apache Flink. Edite los roles de servicio solo cuando Managed Service para Apache Flink proporcione orientación para hacerlo.

Roles vinculados a servicios para Managed Service para Apache Flink

Compatible con roles vinculados al servicio	Sí
---	----

Un rol vinculado al servicio es un tipo de rol de servicio que está vinculado a un Servicio de AWS. El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados a servicios aparecen en la Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulte [Servicios de AWS que funcionan con IAM](#). Busque un servicio en la tabla que incluya Yes en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Ejemplos de políticas basadas en identidades de Amazon Managed Service para Apache Flink

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Managed Service para Apache Flink. Tampoco pueden realizar tareas mediante la AWS Management Console, la AWS Command Line Interface (AWS CLI) o la API de AWS. Para conceder permiso a los usuarios para realizar acciones en los recursos que necesiten, un administrador de IAM puede crear políticas de IAM. A continuación, el administrador puede añadir las políticas de IAM a roles, y los usuarios pueden asumirlos.

Para obtener información sobre cómo crear una política basada en identidad de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte [Creación de políticas de IAM](#) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Managed Service para Apache Flink, incluido el formato de los ARN para cada tipo de recurso, consulte [Acciones, recursos y claves de condición de Amazon Managed Service para Apache Flink](#) en la Referencia de autorizaciones de servicio.

Temas

- [Prácticas recomendadas sobre las políticas](#)
- [Uso de la consola de Managed Service para Apache Flink](#)
- [Cómo permitir a los usuarios consultar sus propios permisos](#)

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de Managed Service para Apache Flink de la cuenta. Estas acciones pueden generar costes adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas administradas de AWS y continúe con los permisos de privilegio mínimo: a fin de comenzar a conceder permisos a los usuarios y las cargas de trabajo, utilice las políticas administradas de AWS, que conceden permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Se recomienda definir políticas administradas por el cliente de AWS específicas para los casos de uso a fin de reducir aún más los permisos. Con

el fin de obtener más información, consulte las [políticas administradas por AWS](#) o las [políticas administradas por AWS para funciones de trabajo](#) en la Guía del usuario de IAM.

- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte [Políticas y permisos en IAM](#) en la Guía de usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puede usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado, como por ejemplo AWS CloudFormation. Para obtener más información, consulte [Elementos de la política de JSON de IAM: Condición](#) en la Guía del usuario de IAM.
- Utilice 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. xPara más información, consulte la [política de validación del Analizador de acceso de IAM](#) en la Guía de usuario de IAM.
- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesita usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para obtener más información, consulte [Configuración de acceso a una API protegida por MFA](#) en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las [Prácticas recomendadas de seguridad en IAM](#) en la Guía del usuario de IAM.

Uso de la consola de Managed Service para Apache Flink

Para acceder a la consola de Amazon Managed Service para Apache Flink, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle registrar y consultar los detalles acerca de los recursos de Managed Service para Apache Flink en su 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 conceda permisos mínimos para la consola a los usuarios que solo realizan llamadas a la AWS CLI o a la API de AWS. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para asegurarse de que los usuarios y los roles puedan seguir utilizando la consola de Managed Service para Apache Flink, asocie también el `ConsoleAccess` del Servicio administrado para Apache Flink o la política administrada a las entidades `ReadOnly` AWS. 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 administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para realizar esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam: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": "*"
}
]
```

Solución de problemas de identidad y acceso de Amazon Managed Service para Apache Flink

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con Managed Service para Apache Flink e IAM.

Temas

- [No tengo autorización para realizar una acción en Managed Service para Apache Flink](#)
- [No estoy autorizado a realizar iam: PassRole](#)
- [Deseo permitir que personas ajenas a mi cuenta de AWS puedan acceder a mis recursos de Managed Service para Apache Flink](#)

No tengo autorización para realizar una acción en Managed Service para Apache Flink

Si la AWS Management Console le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir 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 Kinesis Analytics:*GetWidget*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis Analytics: 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 Kinesis Analytics:*GetWidget*.

No estoy autorizado a realizar iam: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, se deben actualizar las políticas a fin de permitirle pasar un rol a Managed Service para Apache Flink.

Algunos Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Managed Service para Apache Flink. Sin embargo, la acción requiere que el servicio cuente con permisos que concede 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 de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Deseo permitir que personas ajenas a mi cuenta de AWS puedan acceder a mis recursos de Managed Service para Apache Flink

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si Managed Service para Apache Flink es compatible con estas características, consulte [Cómo funciona Amazon Managed Service para Apache Flink con IAM](#).
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte [Proporcionar acceso a un usuario de IAM a otra Cuenta de AWS de la que es propietario](#) en la Guía del usuario de IAM.

- Para obtener información acerca de cómo proporcionar acceso a tus recursos a Cuentas de AWS de terceros, consulte [Proporcionar acceso a Cuentas de AWS que son propiedad de terceros](#) en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante federación de identidades, consulte [Proporcionar acceso a usuarios autenticados externamente \(federación de identidades\)](#) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte [Cómo los roles de IAM difieren de las políticas basadas en recursos](#) en la Guía del Usuario de IAM.

Prevención de la sustitución confusa entre servicios

En AWS, la suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para actuar en función de los recursos de otro cliente a pesar de que no debe tener los permisos adecuados, lo que lleva a cabo un problema de suplente confuso.

Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta. Esta sección se centra en la prevención de problemas de suplentes confusos entre servicios específica de Managed Service para Apache Flink; sin embargo, puede obtener más información sobre este tema en la sección [El problema del suplente confuso](#) de la Guía del usuario de IAM.

En el contexto del servicio gestionado para Apache Flink, le recomendamos que utilice las claves de contexto `aws: SourceArn` y `aws: SourceAccount` global condition en su política de confianza de roles para limitar el acceso al rol únicamente a las solicitudes generadas por los recursos esperados.

Utilice `aws: SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utilice `aws: SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

El valor de `aws: SourceArn` debe ser el ARN del recurso utilizado por Managed Service para Apache Flink, que se especifica con el siguiente formato:
`arn:aws:kinesisanalytics:region:account:resource.`

La forma más eficaz de protegerse contra el problema del suplente confuso es utilizar la clave de contexto de condición global de `aws: SourceArn` con el ARN completo del recurso.

Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo: `arn:aws:kinesisanalytics::111122223333:*`.

Las políticas de funciones que proporcione a Managed Service para Apache Flink, así como las políticas de confianza de las funciones generadas para usted, pueden utilizar estas claves.

Para protegerse contra el problema de suplente confuso, lleve a cabo los siguientes pasos:

Protección contra el problema del suplente confuso

1. Inicie sesión en AWS Management Console y abra la consola IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Roles y, a continuación, seleccione la tabla que desee modificar.
3. Elija Editar la política de confianza.
4. En la página Editar política de confianza, sustituya la política JSON predeterminada por una política que utilice una o ambas claves contextuales `aws:SourceArn` y `aws:SourceAccount` de condición global. Vea el siguiente ejemplo de política:
5. Elija Actualizar política.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "Account ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:kinesisanalytics:us-east-1:123456789012:application/my-app"
        }
      }
    }
  ]
}
```

}

Supervisión de Managed Service para Apache Flink

Managed Service para Apache Flink proporciona funcionalidad de monitorización para aplicaciones. Para obtener más información, consulte [Registro y supervisión](#).

Validación de la conformidad de Amazon Managed Service para Apache Flink

Audidores externos evalúan la seguridad y la conformidad de Amazon Managed Service para Apache Flink como parte de varios programas de conformidad de AWS. Esto incluye SOC, PCI, HIPAA y otros.

Para obtener una lista de AWS servicios en el ámbito de programas de cumplimiento específicos, consulte . Para obtener información general, consulte [Programas de conformidad de AWS](#).

Puede descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte [Descarga de informes en AWS Artifact](#).

Su responsabilidad en relación con la conformidad al utilizar los servicios de Managed Service para Apache Flink depende de la confidencialidad de los datos, los objetivos de conformidad de su empresa y de la legislación y los reglamentos aplicables. Si su uso de Managed Service para Apache Flink está sujeto a conformidad con normas tales como HIPAA o PCI, AWS proporciona recursos para ayudarle:

- [Guías de inicio rápido de seguridad y conformidad](#): estas guías de implementación tratan consideraciones sobre arquitectura y ofrecen pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- [Arquitectura de seguridad de HIPAA y cumplimiento de Amazon Web Services](#) En este documento técnico, se describe cómo las empresas pueden utilizar AWS para crear aplicaciones conformes con HIPAA.
- [AWS Recursos de conformidad](#): este conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- [AWS Config](#): este servicio de AWS evalúa en qué medida las configuraciones de los recursos cumplen las prácticas internas, las directrices del sector y la normativa.

- [AWS Security Hub](#): este servicio de AWS proporciona una vista integral de su estado de seguridad en AWS que lo ayuda a verificar la conformidad con los estándares y las prácticas recomendadas del sector de seguridad.

FedRAMP

El programa de conformidad con FedRAMP de AWS incluye a Managed Service para Apache Flink como servicio autorizado para FedRAMP. Si es un cliente federal o comercial, puede utilizar el servicio para procesar y almacenar cargas de trabajo confidenciales en el límite de autorización de la región AWS GovCloud (EE. UU.) con datos hasta un nivel de alto impacto, así como en las regiones EE. UU. Este (Norte de Virginia), EE. UU. Este (Ohio), EE. UU. Oeste (Norte de California) y EE. UU. Oeste (Oregón) con datos hasta un nivel moderado.

Puede solicitar acceso a los paquetes de seguridad de FedRAMP de AWS a través del PMO de FedRAMP o de su administrador de cuenta de ventas de AWS. También pueden descargarse a través de AWS Artifact en [AWS Artifact](#).

Para obtener más información, consulte [FedRAMP](#).

Resiliencia en Amazon Managed Service para Apache Flink

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte [Infraestructura global de AWS](#).

Además de la infraestructura global de AWS, Managed Service para Apache Flink ofrece varias características que lo ayudan con sus necesidades de resiliencia y copia de seguridad de los datos.

Recuperación de desastres

Managed Service para Apache Flink se ejecuta en un modo sin servidor y se ocupa de la reducción del rendimiento del host, la disponibilidad de las zonas de disponibilidad y otros problemas

relacionados con la infraestructura realizando una migración automática. Managed Service para Apache Flink logra esto a través de múltiples mecanismos redundantes. Cada aplicación de Managed Service para Apache Flink se ejecuta en un clúster Apache Flink de un solo inquilino. El clúster de Apache Flink se ejecuta JobManager en modo de alta disponibilidad mediante Zookeeper en varias zonas de disponibilidad. Managed Service para Apache Flink implementa Apache Flink utilizando Amazon EKS. En Amazon EKS se utilizan varios pods de Kubernetes para cada región AWS y en todas las zonas de disponibilidad. En caso de producirse un error, Managed Service para Apache Flink primero intenta recuperar la aplicación dentro del clúster de Apache Flink en ejecución utilizando los puntos de control de la aplicación, si están disponibles.

Managed Service para Apache Flink realiza una copia de seguridad del estado de la aplicación mediante puntos de control e instantáneas:

- Los puntos de control son copias de seguridad del estado de la aplicación que Managed Service para Apache Flink crea automáticamente de forma periódica y que utiliza para restaurarlas en caso de errores.
- Las instantáneas son copias de seguridad del estado de la aplicación que se crean y restauran manualmente.

Para obtener más información sobre los puntos de comprobación y las instantáneas, consulte [Tolerancia a errores](#).

Control de versiones

Las versiones almacenadas del estado de la aplicación se versionan de la siguiente manera:

- El servicio versiona automáticamente los puntos de control. Si el servicio utiliza un punto de control para reiniciar la aplicación, se utilizará el último punto de control.
- Los puntos guardados se versionan utilizando el parámetro de la acción.
SnapshotName [CreateApplicationSnapshot](#)

Managed Service para Apache Flink cifra los datos almacenados en puntos de control y puntos de almacenamiento.

Seguridad de la infraestructura en Amazon Managed Service para Apache Flink

Dado que se trata de un servicio administrado, Managed Service para Apache Flink está protegido por los procedimientos de seguridad de red globales de AWS, que se describen en el documento técnico [Amazon Web Services: Información general acerca de los procesos de seguridad](#).

Puede utilizar llamadas a la API publicadas en AWS para acceder a Managed Service para Apache Flink a través de la red. Todas las llamadas de API al servicio gestionado de Apache Flink se protegen mediante Seguridad de la capa de transporte (TLS) y se autentican mediante IAM. Los clientes deben ser compatibles con TLS 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). 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 puede utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Mejores prácticas de seguridad para el servicio gestionado de Managed Service para Apache Flink

Amazon Managed Service para Apache Flink 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 suponen 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.

Implementación del acceso a los privilegios mínimos

Cuando concede permisos, debe decidir a quién concede cada permiso y para qué recurso de Managed Service para Apache Flink se lo concede. Habilite las acciones específicas que desea permitir en dichos recursos. Por lo tanto, debe conceder únicamente los permisos obligatorios para realizar una tarea. La implementación del acceso con privilegios mínimos es esencial a la hora de reducir los riesgos de seguridad y el impacto que podrían causar los errores o los intentos malintencionados.

Uso de los roles de IAM para obtener acceso a otros servicios de Amazon

La aplicación Managed Service para Apache Flink debe tener credenciales válidas para poder obtener acceso a recursos en otros servicios, como los flujos de datos de Kinesis, los flujos de Kinesis Data Firehose o los buckets de Amazon S3. No debe almacenar las credenciales de AWS directamente en la aplicación ni en un bucket de Amazon S3. Estas son las credenciales a largo plazo que no rotan automáticamente y que podrían tener un impacto empresarial significativo si se comprometen.

En su lugar, debería utilizar un rol de IAM para administrar las credenciales temporales que la aplicación utiliza para obtener acceso a otros recursos. Al utilizar un rol, no tiene que utilizar credenciales a largo plazo para acceder a otros recursos.

Para obtener más información, consulte los siguientes temas de la guía del usuario de IAM:

- [Roles de IAM](#)
- [Situaciones habituales con los roles: usuarios, aplicaciones y servicios](#)

Implementación del cifrado en el servidor en recursos dependientes

Los datos en reposo y los datos en tránsito se cifran en Managed Service para Apache Flink y este cifrado no se puede deshabilitar. Debe implementar el cifrado del lado del servidor en los recursos dependientes, como los flujos de datos de Kinesis, los flujos de Kinesis Data Firehose y los buckets de Amazon S3. Para obtener más información acerca de la implementación del cifrado del lado del servidor en recursos dependientes, consulte [Protección de los datos](#).

Se utiliza para supervisar las llamadas CloudTrail a la API

Managed Service para Apache Flink está integrado con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, rol o servicio de Amazon en Managed Service para Apache Flink.

Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Managed Service for Apache Flink, la dirección IP desde la que se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información, consulte [the section called “Uso de AWS CloudTrail”](#).

Registro y supervisión en Amazon Managed Service para Apache Flink

El monitoreo es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de las aplicaciones Managed Service para Apache Flink. Debe recopilar los datos de monitoreo de todas las partes de la solución de AWS para que le resulte más sencillo depurar un error que se produce en distintas partes del código, en caso de que ocurra.

Antes de empezar a monitorear Managed Service para Apache Flink, debe crear un plan de monitoreo que incluya respuestas a las siguientes preguntas:

- ¿Cuáles son los objetivos de la supervisión?
- ¿Qué recursos va a supervisar?
- ¿Con qué frecuencia va a supervisar estos recursos?
- ¿Qué herramientas de monitoreo va a utilizar?
- ¿Quién se encargará de realizar las tareas de monitoreo?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso consiste en establecer un punto de referencia para el rendimiento normal de Managed Service para Apache Flink en su entorno. Para ello, mida el rendimiento en diversas ocasiones y con diferentes condiciones de carga. A medida que supervise Managed Service para Apache Flink, puede almacenar datos de monitoreo históricos. Luego, puede compararlos con los datos de rendimiento actuales, identificar patrones de rendimiento normal y anomalías en el rendimiento, así como desarrollar métodos para la resolución de problemas.

Temas

- [Registro](#)
- [Monitorización](#)
- [Configuración del registro de aplicaciones](#)
- [Análisis de registros con CloudWatch Logs Insights](#)
- [Visualización de métricas y dimensiones en Managed Service para Apache Flink](#)
- [Escribir mensajes personalizados en los CloudWatch registros](#)
- [Registro de llamadas de API de Managed Service para Apache Flink con AWS CloudTrail](#)

Registro

El registro es importante para que las aplicaciones de producción comprendan los errores y las fallas. Sin embargo, el subsistema de registro necesita recopilar y reenviar las entradas de registro a CloudWatch Logs. Si bien algunos registros están bien y son deseables, los registros extensos pueden sobrecargar el servicio y provocar un retraso en la aplicación Flink. Registrar las excepciones y advertencias es, sin duda, una buena idea. Sin embargo, no puede generar un mensaje de registro para todos y cada uno de los mensajes que procesa la aplicación Flink. Flink está optimizado para un rendimiento alto y una latencia baja, pero el subsistema de registro no lo está. En caso de que realmente sea necesario generar un registro de salida para cada mensaje procesado, utilice uno adicional DataStream dentro de la aplicación Flink y un receptor adecuado para enviar los datos a Amazon S3 o CloudWatch. No utilice el sistema de registro de Java para este propósito. Además, la configuración de Debug Monitoring Log Level de Managed Service para Apache Flink genera una gran cantidad de tráfico, lo que puede generar retrasos. Solo debe usarlo mientras investiga activamente los problemas de la aplicación.

Consulta de registros con Logs Insights CloudWatch

CloudWatch Logs Insights es un potente servicio para consultar registros a escala. Los clientes deben aprovechar sus capacidades para buscar rápidamente en los registros a fin de identificar y mitigar los errores durante los eventos operativos.

La siguiente consulta busca las excepciones en todos los registros del administrador de tareas y las ordena según la hora en que se produjeron.

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
  @message like /(Error|Exception)/
| sort @timestamp desc
```

Para ver otras consultas útiles, consulte [Consultas de ejemplo](#).

Monitorización

Al ejecutar aplicaciones de streaming en producción, se propone ejecutar la aplicación de forma continua e indefinida. Es crucial implementar un monitoreo y una alarma adecuados de todos los componentes, no solo de la aplicación Flink. De lo contrario, se corre el riesgo de pasar por alto los problemas emergentes desde el principio y solo darse cuenta de un problema operativo una vez que

se esté resolviendo por completo y sea mucho más difícil de mitigar. Entre los aspectos generales que deben supervisarse se incluyen:

- ¿El origen ingiere datos?
- ¿Los datos se leen desde el origen (desde la perspectiva del origen)?
- ¿La aplicación Flink recibe datos?
- ¿La aplicación Flink puede mantener el ritmo o se está quedando atrás?
- ¿La aplicación Flink conserva los datos en el receptor (desde la perspectiva de la aplicación)?
- ¿El receptor recibe datos?

Por lo tanto, deberían considerarse métricas más específicas para la aplicación Flink. Este [CloudWatch panel](#) proporciona un buen punto de partida. Para obtener más información sobre las métricas que deben supervisarse para las aplicaciones de producción, consulte [Uso de CloudWatch alarmas con Amazon Managed Service para Apache Flink](#). Estas métricas incluyen:

- `records_lag_max` y `millisBehindLatest`: si la aplicación consume contenido de Kinesis o Kafka, estas métricas indican si la aplicación se está retrasando y es necesario escalarla para poder soportar la carga actual. Se trata de una buena métrica genérica y fácil de rastrear para todo tipo de aplicaciones. Sin embargo, solo se puede usar para el escalado reactivo, es decir, cuando la aplicación ya se ha retrasado.
- `CPUUtilization` `heapMemoryUtilization`: estas métricas proporcionan una buena indicación de la utilización general de los recursos de la aplicación y se pueden utilizar para un escalado proactivo, a menos que la aplicación esté vinculada a la E/S.
- tiempo de inactividad: un tiempo de inactividad superior a cero indica que la aplicación ha fallado. Si el valor es mayor que 0, la aplicación no está procesando ningún dato.
- `lastCheckpointSize` `lastCheckpointDuration`: estas métricas controlan la cantidad de datos que se almacenan en el estado y el tiempo que se tarda en pasar por un punto de control. Si los puntos de control aumentan o tardan mucho, la aplicación dedica tiempo continuamente a los puntos de control y tiene menos ciclos de procesamiento real. En algunos puntos, los puntos de control pueden crecer demasiado o tardar tanto que fallan. Además de monitorear los valores absolutos, los clientes también deberían considerar monitorear la tasa de cambio con `RATE(lastCheckpointSize)` y `RATE(lastCheckpointDuration)`.
- `numberOfFailedPuntos de control`: esta métrica cuenta el número de puntos de control fallidos desde que se inició la aplicación. En función de la aplicación, puede ser tolerable que los puntos de control fallen ocasionalmente. Sin embargo, si los puntos de control fallan con regularidad,

es probable que la aplicación no esté en buen estado y necesite más atención. Recomendamos monitorizar `RATE(numberOfFailedCheckpoints)` para avisar sobre el gradiente y no sobre los valores absolutos.

Configuración del registro de aplicaciones

Al añadir una opción de CloudWatch registro de Amazon a su aplicación Managed Service for Apache Flink, puede supervisar los eventos de la aplicación o los problemas de configuración.

En este tema se describe cómo configurar la aplicación para escribir los eventos de la aplicación en una transmisión de CloudWatch registros. Una opción de CloudWatch registro es un conjunto de ajustes y permisos de la aplicación que la aplicación utiliza para configurar la forma en que escribe los eventos de la aplicación en los CloudWatch registros. Puede agregar y configurar una opción de CloudWatch registro mediante la tecla () AWS Management Console o la tecla AWS Command Line Interface (AWS CLI).

Tenga en cuenta lo siguiente sobre la adición de una opción de CloudWatch registro a su aplicación:

- Al añadir una opción de CloudWatch registro mediante la consola, Managed Service for Apache Flink crea el grupo de CloudWatch registros y el flujo de registros automáticamente y añade los permisos que la aplicación necesita para escribir en el flujo de registros.
- Al añadir una opción de CloudWatch registro mediante la API, también debe crear el grupo de registros y el flujo de registros de la aplicación, además de añadir los permisos que la aplicación necesita para escribir en el flujo de registros.

Este tema contiene las siguientes secciones:

- [Configuración del CloudWatch registro mediante la consola](#)
- [Configuración del CloudWatch registro mediante la CLI](#)
- [Niveles de monitoreo de aplicaciones](#)
- [Prácticas recomendadas de registro](#)
- [Solución de problemas de registro](#)
- [Paso siguiente](#)

Configuración del CloudWatch registro mediante la consola

Al habilitar el CloudWatch registro de la aplicación en la consola, se crean CloudWatch automáticamente un grupo de registros y un flujo de registros. Además, la política de permisos de la aplicación se actualiza con los permisos para escribir en la transmisión.

Managed Service for Apache Flink crea un grupo de registros denominado según la siguiente convención, donde *ApplicationName* es el nombre de la aplicación.

```
/AWS/KinesisAnalytics/ApplicationName
```

Managed Service para Apache Flink crea un flujo de registros en el nuevo grupo de registros con el siguiente nombre.

```
kinesis-analytics-log-stream
```

Para establecer el nivel de métricas de supervisión de la aplicación y el nivel de registro de supervisión, utilice la sección del nivel de registro de supervisión de la página de configuración de la aplicación. Para obtener más información acerca de los niveles del flujo de la aplicación, consulte [the section called “Niveles de monitoreo de aplicaciones”](#).

Configuración del CloudWatch registro mediante la CLI

Para añadir una opción de CloudWatch registro mediante el AWS CLI, haga lo siguiente:

- Cree un grupo de CloudWatch registros y un flujo de registros.
- Añada una opción de registro al crear una aplicación mediante la [CreateApplication](#) acción o añada una opción de registro a una aplicación existente mediante la [AddApplicationCloudWatchLoggingOption](#) acción.
- Añada permisos a la política de la aplicación para escribir en los registros.

Esta sección contiene los siguientes temas:

- [Crear un grupo de CloudWatch registros y un flujo de registros](#)
- [Cómo trabajar con las opciones de CloudWatch registro de aplicaciones](#)
- [Agregar permisos para escribir en el flujo de CloudWatch registro](#)

Crear un grupo de CloudWatch registros y un flujo de registros

Puede crear un grupo de CloudWatch registros y transmitirlos mediante la consola de CloudWatch registros o la API. Para obtener información sobre cómo crear un grupo de CloudWatch registros y un flujo de registros, consulte [Trabajar con grupos de registros y flujos de registros](#).

Cómo trabajar con las opciones de CloudWatch registro de aplicaciones

Utilice las siguientes acciones de la API para añadir una opción de CloudWatch registro a una aplicación nueva o existente o para cambiar una opción de registro de una aplicación existente. Para obtener información sobre cómo utilizar un archivo JSON como entrada para una acción de API, consulte [Ejemplo de código de API de Managed Service para Apache Flink](#).

Añadir una opción de CloudWatch registro al crear una aplicación

En el siguiente ejemplo, se muestra cómo utilizar la `CreateApplication` acción para añadir una opción de CloudWatch registro al crear una aplicación. En el ejemplo, sustituya el *nombre de recurso de Amazon (ARN) del flujo de CloudWatch registro para añadirlo a la nueva aplicación* con su propia información. Para obtener más información sobre la acción, consulte [CreateApplication](#).

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "test-application-description",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
          "FileKey": "myflink.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    }
  },
  "CloudWatchLoggingOptions": [{
    "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add to the new application>"
  }]
}
```

Añadir una opción de CloudWatch registro a una aplicación existente

En el siguiente ejemplo, se muestra cómo utilizar la `AddApplicationCloudWatchLoggingOption` acción para añadir una opción de CloudWatch registro a una aplicación existente. En el siguiente ejemplo, reemplace cada *marcador de posición del usuario* con su propia información. Para obtener más información sobre la acción, consulte [AddApplicationCloudWatchLoggingOption](#).

```
{
  "ApplicationName": "<Name of the application to add the log option to>",
  "CloudWatchLoggingOption": {
    "LogStreamARN": "<ARN of the log stream to add to the application>"
  },
  "CurrentApplicationVersionId": <Version of the application to add the log to>
}
```

Actualización de una opción de CloudWatch registro existente

En el siguiente ejemplo, se muestra cómo utilizar la `UpdateApplication` acción para modificar una opción de CloudWatch registro existente. En el siguiente ejemplo, reemplace cada *marcador de posición del usuario* con su propia información. Para obtener más información sobre la acción, consulte [UpdateApplication](#).

```
{
  "ApplicationName": "<Name of the application to update the log option for>",
  "CloudWatchLoggingOptionUpdates": [
    {
      "CloudWatchLoggingOptionId": "<ID of the logging option to modify>",
      "LogStreamARNUpdate": "<ARN of the new log stream to use>"
    }
  ],
  "CurrentApplicationVersionId": <ID of the application version to modify>
}
```

Eliminar una opción de CloudWatch registro de una aplicación

En el siguiente ejemplo, se muestra cómo utilizar la `DeleteApplicationCloudWatchLoggingOption` acción para eliminar una opción de

CloudWatch registro existente. En el siguiente ejemplo, reemplace cada *marcador de posición del usuario* con su propia información. Para obtener más información sobre la acción, consulte [DeleteApplicationCloudWatchLoggingOption](#).

```
{
  "ApplicationName": "<Name of application to delete log option from>",
  "CloudWatchLoggingOptionId": "<ID of the application log option to delete>",
  "CurrentApplicationVersionId": <Version of the application to delete the log option
  from>
}
```

Configuración del nivel de registro de la aplicación

Para establecer el nivel de registro de la aplicación, utilice el parámetro [MonitoringConfiguration](#) de la acción [CreateApplication](#) o el parámetro [MonitoringConfigurationUpdate](#) de la acción [UpdateApplication](#).

Para obtener más información acerca de los niveles del flujo de la aplicación, consulte [the section called "Niveles de monitoreo de aplicaciones"](#).

Establezca el nivel de registro de la aplicación al crear una aplicación

El siguiente ejemplo de solicitud de la acción [CreateApplication](#) establece el nivel de registro de la aplicación en INFO.

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My Application Description",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "FlinkApplicationConfiguration":
```

```

    "MonitoringConfiguration": {
      "ConfigurationType": "CUSTOM",
      "LogLevel": "INFO"
    }
  },
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}

```

Actualizar el nivel de registro de la aplicación

El siguiente ejemplo de solicitud de la acción [UpdateApplication](#) establece el nivel de registro de la aplicación en INFO.

```

{
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "MonitoringConfigurationUpdate": {
        "ConfigurationTypeUpdate": "CUSTOM",
        "LogLevelUpdate": "INFO"
      }
    }
  }
}

```

Agregar permisos para escribir en el flujo de CloudWatch registro

El servicio gestionado de Apache Flink necesita permisos para escribir errores de configuración en él. CloudWatch Puede añadir estos permisos al rol AWS Identity and Access Management (IAM) que asume Managed Service para Apache Flink.

Para obtener más información sobre el uso de un rol de IAM para Managed Service para Apache Flink, consulte [Identidad y gestión de acceso para Amazon Managed Service para Apache Flink](#).

Política de confianza

Para conceder permisos a Managed Service para Apache Flink para asumir un rol de IAM, puede asignar la siguiente política de confianza al rol.

```

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

```

```

"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "kinesisanalytics.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
]
}

```

Política de permisos

Para conceder permisos a una aplicación para escribir eventos de registro CloudWatch desde un recurso de Managed Service for Apache Flink, puede utilizar la siguiente política de permisos de IAM. Proporcione los nombres de recursos de Amazon (ARN) para el grupo de registros y flujo.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt0123456789000",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:log-stream:my-log-stream*",
        "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:*",
        "arn:aws:logs:us-east-1:123456789012:log-group:*"
      ]
    }
  ]
}

```

Niveles de monitoreo de aplicaciones

Usted controla la generación de los mensajes de registro de la aplicación mediante el nivel de métricas de monitoreo y el nivel de registro de monitoreo de la aplicación.

El nivel de métricas de supervisión de la aplicación controla la granularidad de los mensajes de registro. Los niveles de las métricas de monitoreo se definen de la siguiente manera:

- **Aplicación:** las métricas se aplican a toda la aplicación.
- **Tarea:** las métricas se ajustan al alcance de cada tarea. Para obtener más información acerca de las tareas, consulte [the section called “Escalado”](#).
- **Operador:** se determina el alcance de las métricas para cada operador. Para obtener más información sobre los operadores, consulte [the section called “Operadores de la API de DataStream”](#).
- **Paralelismo:** las métricas se basan en el paralelismo de la aplicación. Solo puede establecer este nivel de métricas mediante el [MonitoringConfigurationUpdate](#) parámetro de la [UpdateApplication](#) API. No puede establecer este nivel de métricas mediante la consola. Consulte [the section called “Escalado”](#) para obtener información sobre el paralelismo.

El nivel de registro de supervisión de la aplicación controla la verbosidad del registro de la aplicación. Los niveles de registro de monitoreo se definen de la siguiente manera:

- **Error:** posibles eventos catastróficos de la aplicación.
- **Advertencia:** situaciones potencialmente dañinas de la aplicación.
- **Información:** eventos de fallo informativos y transitorios de la aplicación. Recomendamos utilizar este nivel de registro.
- **Depuración:** eventos informativos detallados que son muy útiles para depurar una aplicación. Nota: utilice este nivel únicamente con fines de depuración temporal.

Prácticas recomendadas de registro

Se recomienda que la aplicación utilice el nivel de registro de información. Recomendamos este nivel para asegurarse de que aparecen los errores de Apache Flink, que se registran en el nivel de información y no en el nivel de error.

Le recomendamos que utilice el nivel de depuración solo de forma temporal mientras investiga los problemas de las aplicaciones. Vuelva al nivel de información cuando se resuelva el problema. El uso del nivel de registro de depuración afectará significativamente al rendimiento de la aplicación.

El registro excesivo también puede afectar significativamente al rendimiento de la aplicación. Por ejemplo, le recomendamos que no escriba una entrada de registro para cada registro procesado. El

registro excesivo puede provocar graves cuellos de botella en el procesamiento de datos y provocar retrasos a la hora de leer los datos de las fuentes.

Solución de problemas de registro

Si los registros de la aplicación no se escriben en el flujo de registro, compruebe lo siguiente:

- Compruebe que las políticas y el rol de IAM de su aplicación sean correctos. La política de su aplicación necesita los siguientes permisos para acceder a su flujo de registro:
 - `logs:PutLogEvents`
 - `logs:DescribeLogGroups`
 - `logs:DescribeLogStreams`

Para obtener más información, consulte [the section called “Agregar permisos para escribir en el flujo de CloudWatch registro”](#).

- Comprobar que la aplicación se esté ejecutando. Para comprobar el estado de la aplicación, consulta la página de la aplicación en la consola o usa las [ListApplications](#) acciones [DescribeApplication](#).
- Supervise CloudWatch las métricas, por ejemplo, `downtime` para diagnosticar otros problemas de la aplicación. Para obtener información sobre la lectura de CloudWatch las métricas, consulte [Métricas y dimensiones en Managed Service para Apache Flink](#).

Paso siguiente

Una vez que haya habilitado el CloudWatch registro en su aplicación, puede usar CloudWatch Logs Insights para analizar los registros de la aplicación. Para obtener más información, consulte [the section called “Análisis de registros”](#).

Análisis de registros con CloudWatch Logs Insights

Una vez que haya agregado una opción de CloudWatch registro a su aplicación, tal como se describe en la sección anterior, puede usar CloudWatch Logs Insights para consultar sus flujos de registro en busca de eventos o errores específicos.

CloudWatch Logs Insights te permite buscar y analizar de forma interactiva tus datos de registro en CloudWatch Logs.

Para obtener información sobre cómo empezar a utilizar CloudWatch Logs Insights, consulte [Analizar los datos de registro con CloudWatch Logs Insights](#).

Ejecutar una consulta de muestra

En esta sección se describe cómo ejecutar un ejemplo de consulta de CloudWatch Logs Insights.

Requisitos previos

- Los grupos de registros y los flujos de registros existentes se configuran en CloudWatch Logs.
- Los registros existentes se almacenan en CloudWatch los registros.

Si utilizas servicios como AWS CloudTrail Amazon Route 53 o Amazon VPC, probablemente ya hayas configurado los registros de esos servicios para que vayan a CloudWatch Logs. Para obtener más información sobre el envío de CloudWatch registros a Logs, consulte [Introducción a CloudWatch Logs](#).

Las consultas en CloudWatch Logs Insights devuelven un conjunto de campos de eventos de registro o el resultado de una agregación matemática u otra operación realizada en el registro de eventos. Esta sección muestra una consulta que devuelve una lista de eventos de registro.

Para ejecutar una consulta de ejemplo CloudWatch de Logs Insights

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En el panel de navegación, elija Información.
3. El editor de consultas cerca de la parte superior de la pantalla contiene una consulta predeterminada que devuelve los 20 eventos de registro más recientes. Encima del editor de consultas, seleccione un grupo de registros que se va a consultar.

Al seleccionar un grupo de CloudWatch registros, Logs Insights detecta automáticamente los campos de los datos del grupo de registros y los muestra en Campos detectados en el panel derecho. También muestra un gráfico de barras de eventos de registro en este grupo de registros con el paso del tiempo. Este gráfico de barras muestra la distribución de los eventos en el grupo de registros que coincide con la consulta y el intervalo de tiempo, no solo los eventos que se muestran en la tabla.

4. Elija Ejecutar consulta.

Aparecen los resultados de la consulta. En este ejemplo, los resultados son los últimos 20 eventos de registro de cualquier tipo.

5. Para ver todos los campos para uno de los eventos de registro devueltos, seleccione la flecha que aparece a la izquierda de ese evento de registro.

Para obtener más información sobre cómo ejecutar y modificar las consultas de CloudWatch Logs Insights, consulte [Ejecutar y modificar una consulta de muestra](#).

Consultas de ejemplo

Esta sección contiene consultas de ejemplo de CloudWatch Logs Insights para analizar los registros de aplicaciones de Managed Service for Apache Flink. Estas consultas buscan varios ejemplos de condiciones de error y sirven como plantillas para escribir consultas que encuentren otras condiciones de error.

Note

Sustituya la región (*us-west-2*), el identificador de cuenta (*012345678901*) y el nombre de la aplicación (*YourApplication*) en los siguientes ejemplos de consultas por la región de la aplicación y el identificador de cuenta.

Este tema contiene las siguientes secciones:

- [Operaciones de análisis: distribución de las tareas](#)
- [Operaciones de análisis: cambio en el paralelismo](#)
- [Errores de análisis: acceso denegado](#)
- [Errores de análisis: no se encontró la fuente o el receptor](#)
- [Errores de análisis: errores relacionados con las tareas de la aplicación](#)

Operaciones de análisis: distribución de las tareas

La siguiente consulta de CloudWatch Logs Insights devuelve el número de tareas que el administrador de tareas de Apache Flink distribuye entre los administradores de tareas. Debe configurar el período de tiempo de la consulta para que coincida con la ejecución de un trabajo,

de modo que la consulta no devuelva tareas de trabajos anteriores. Para obtener más información acerca del Paralelismo, consulte [Escalado](#).

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

La siguiente consulta de CloudWatch Logs Insights devuelve las subtareas asignadas a cada administrador de tareas. La cantidad total de subtareas es la suma del paralelismo de cada tarea. El paralelismo de las tareas se deriva del paralelismo de los operadores y, de forma predeterminada, es el mismo que el paralelismo de la aplicación, a menos que se modifique en el código especificando `setParallelism`. Para obtener información sobre cómo configurar el paralelismo de operadores, consulte [Configuración del paralelismo: nivel del operador](#) en la [documentación de Apache Flink](#).

```
fields @timestamp, @tmid, @subtask
| filter message like /Deploying/
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
| limit 2000
```

Para obtener más información acerca de la programación de tareas, consulte [Trabajos y programación](#) en la [documentación de Apache Flink](#).

Operaciones de análisis: cambio en el paralelismo

La siguiente consulta de CloudWatch Logs Insights devuelve los cambios en el paralelismo de una aplicación (por ejemplo, debido al escalado automático). Esta consulta también devuelve los cambios manuales en el paralelismo de la aplicación. Para obtener más información sobre el escalado automático, consulte [the section called “Escalado automático”](#).

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Errores de análisis: acceso denegado

La siguiente consulta de CloudWatch Logs Insights devuelve registros. Access Denied

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Errores de análisis: no se encontró la fuente o el receptor

La siguiente consulta CloudWatch de Logs Insights devuelve ResourceNotFound registros. ResourceNotFound los registros se generan si no se encuentra una fuente o un receptor de Kinesis.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

Errores de análisis: errores relacionados con las tareas de la aplicación

La siguiente consulta de CloudWatch Logs Insights devuelve los registros de errores relacionados con las tareas de una aplicación. Estos registros se generan si el estado de una aplicación cambia de RUNNING a RESTARTING.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /switched from RUNNING to RESTARTING/
| sort @timestamp desc
```

En el caso de las aplicaciones que utilizan la versión 1.8.2 de Apache Flink y versiones anteriores, los errores relacionados con las tareas harán que el estado de la aplicación cambie de RUNNING a FAILED. Cuando utilice Apache Flink 1.8.2 y versiones anteriores, utilice la siguiente consulta para buscar errores relacionados con las tareas de la aplicación:

```
fields @timestamp,@message
```

```
| filter applicationARN like /arn:aws:kinesisanalyticsus-  
west-2:012345678901:application\YourApplication/  
| filter @message like /switched from RUNNING to FAILED/  
| sort @timestamp desc
```

Visualización de métricas y dimensiones en Managed Service para Apache Flink

Este tema contiene las siguientes secciones:

- [Métricas de aplicación](#)
- [Métricas del conector de flujo de datos de Kinesis](#)
- [Métricas de Amazon MSK Connector](#)
- [Métricas de Apache Zeppelin](#)
- [Visualización de métricas CloudWatch](#)
- [Configuración de los CloudWatch niveles de informes de métricas](#)
- [Uso de métricas personalizadas con Amazon Managed Service para Apache Flink](#)
- [Uso de CloudWatch alarmas con Amazon Managed Service para Apache Flink](#)

Cuando su Managed Service for Apache Flink procesa una fuente de datos, Managed Service for Apache Flink informa a Amazon de las siguientes métricas y dimensiones. CloudWatch

Métricas de aplicación

Métrica	Unidad	Descripción	Nivel	Notas de uso
backPressureTimeMsPerSecond*	Milisegundos	El tiempo (en milisegundos) que esta tarea u operador tiene retraso por segundo.	Tarea, Operador, Paralelismo	*Disponible para Managed Service para Apache Flink que ejecutan únicamente la versión 1.13 de Flink.

Métrica	Unidad	Descripción	Nivel	Notas de uso
				Estas métricas pueden resultar útiles para identificar los cuellos de botella en una aplicación.
busyTimeMsPerSecond*	Milisegundos	El tiempo (en milisegundos) que esta tarea u operador está ocupado (ni inactivo ni con retraso) por segundo. Puede ser NaN si no se pudo calcular el valor.	Tarea, Operador, Paralelismo	*Disponible para Managed Service para Apache Flink que ejecutan únicamente la versión 1.13 de Flink. Estas métricas pueden resultar útiles para identificar los cuellos de botella en una aplicación.

Métrica	Unidad	Descripción	Nivel	Notas de uso
cpuUtilization	Porcentaje	El porcentaje total de utilización de CPU en los administradores de tareas. Por ejemplo, si hay cinco administradores de tareas, Managed Service para Apache Flink publica cinco muestras de esta métrica por intervalo de informes.	Aplicación	Puede usar esta métrica para monitorear el uso mínimo, promedio y máximo de la CPU en su aplicación. La CPUUtilization métrica solo tiene en cuenta el uso de la CPU del proceso de TaskManager JVM que se ejecuta dentro del contenedor.

Métrica	Unidad	Descripción	Nivel	Notas de uso
container CPUUtilization	Porcentaje	Porcentaje total de uso de la CPU en los contenedores del administrador de tareas del clúster de aplicaciones Flink. Por ejemplo, si hay cinco administradores de tareas, también hay cinco TaskManager contenedores y Managed Service for Apache Flink publica de 2 a cinco muestras de esta métrica por cada intervalo de informe de 1 minuto.	Aplicación	<p>Se calcula por contenedor de la siguiente manera:</p> <p>Tiempo total de CPU (en segundos) consumido por el contenedor * 100 / Límite de CPU del contenedor (en CPU/segundos)</p> <p>La CPUUtilization métrica solo tiene en cuenta el uso de la CPU del proceso de TaskManager JVM que se ejecuta dentro del contenedor. Hay otros componentes que se ejecutan fuera de la JVM dentro del mismo contenedor</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
				r. La métrica <code>containerCPUUtilization</code> ofrece un panorama más completo, que incluye todos los procesos en términos de agotamiento de la CPU en el contenedor y de los fallos resultantes.

Métrica	Unidad	Descripción	Nivel	Notas de uso
containerMemoryUtilization	Porcentaje	Porcentaje general de uso de memoria en los contenedores del administrador de tareas del clúster de aplicaciones Flink. Por ejemplo, si hay cinco administradores de tareas, también hay cinco TaskManagers contenedores y Managed Service for Apache Flink publica de 2 a cinco muestras de esta métrica por cada intervalo de informe de 1 minuto.	Aplicación	<p>Se calcula por contenedor de la siguiente manera:</p> <p>Uso de memoria del contenedor (bytes) * 100 / Límite de memoria del contenedor según las especificaciones de implementación del pod (en bytes)</p> <p>ManagedMemoryUtilizations Las métricas HeapMemoryUtilization y solo tienen en cuenta métricas de memoria específicas, como el uso de memoria dinámica de la TaskManag</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>er JVM o la memoria administrada (uso de memoria fuera de la JVM para procesos nativos, como el backend estatal de RockSDB).</p> <p>La métrica <code>containerMemoryUtilization</code> ofrece una imagen más completa al incluir la memoria del conjunto de trabajo, lo que permite medir mejor el agotamiento total de la memoria. Cuando se agote, pasará al <code>pod. Out of Memory Error TaskManager</code></p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
container DiskUtili zation	Porcentaje	Porcentaje total de uso del disco en los contenedores del administrador de tareas del clúster de aplicaciones Flink. Por ejemplo, si hay cinco administradores de tareas, hay cinco TaskManag er contenedores y Managed Service for Apache Flink publica de 2 a cinco muestras de esta métrica por cada intervalo de informe de 1 minuto.	Aplicación	<p>Se calcula por contenedor de la siguiente manera:</p> <p>Uso del disco en bytes * 100 / Límite del disco por contenedor en bytes</p> <p>En el caso de los contenedores, representa la utilización del sistema de archivos en el que está configurado el volumen raíz del contenedor.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
currentInputWatermark	Milisegundos	La última marca de agua que ha recibido esta aplicación/operador/tarea/subproceso	Aplicación, Operador, Tarea, Paralelismo	Este registro solo se emite para dimensiones con dos entradas. Es el valor mínimo de las últimas marcas de agua recibidas.
currentOutputWatermark	Milisegundos	La última marca de agua que ha enviado esta aplicación/operador/tarea/subproceso	Aplicación, Operador, Tarea, Paralelismo	

Métrica	Unidad	Descripción	Nivel	Notas de uso
downtime	Milisegundos	En el caso de los trabajos que actualmente se encuentran en situación de fallo o recuperación, el tiempo transcurrido durante la interrupción.	Aplicación	Esta métrica mide el tiempo transcurrido mientras un trabajo está fallando o se está recuperando. Esta métrica devuelve 0 para los trabajos en ejecución y -1 para los trabajos completados. Si esta métrica no es 0 o -1, indica que no se pudo ejecutar el trabajo de Apache Flink para la aplicación.

Métrica	Unidad	Descripción	Nivel	Notas de uso
fullRestarts	Recuento	La cantidad total de veces que este trabajo se ha reiniciado por completo desde que fue enviado. Esta métrica no mide los reinicios detallados.	Aplicación	Puede usar esta métrica para evaluar el estado general de la aplicación. Managed Service para Apache Flink puede reiniciar los valores controlados. Más reinicios de lo normal pueden indicar un problema con la aplicación.

Métrica	Unidad	Descripción	Nivel	Notas de uso
heapMemoryUtilization	Porcentaje	Utilización general de la memoria dinámica en los administradores de tareas. Por ejemplo, si hay cinco administradores de tareas, Managed Service para Apache Flink publica cinco muestras de esta métrica por intervalo de informes.	Aplicación	Puede usar esta métrica para monitorear el uso mínimo, promedio y máximo de la memoria dinámica en su aplicación. HeapMemoryUtilization Solo tiene en cuenta métricas de memoria específicas, como el uso de memoria acumulada de la JVM. TaskManager

Métrica	Unidad	Descripción	Nivel	Notas de uso
idleTimeMsPerSecond*	Milisegundos	El tiempo (en milisegundos) por segundo que esta tarea u operador está inactivo (sin datos que procesar) . El tiempo de inactividad no incluye el tiempo de retraso, por lo que si la tarea está retrasada, no está inactiva.	Tarea, Operador, Paralelismo	<p>*Disponible para Managed Service para Apache Flink que ejecutan únicamente la versión 1.13 de Flink.</p> <p>Estas métricas pueden resultar útiles para identificar los cuellos de botella en una aplicación.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
lastCheckpointSize	Bytes	El tamaño total del último punto de control	Aplicación	<p>Puede usar esta métrica para determinar la utilización del almacenamiento de las aplicaciones en ejecución.</p> <p>Si el valor de esta métrica aumenta, esto puede indicar que hay un problema con la aplicación, como una pérdida de memoria o un cuello de botella.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
lastCheckpointDuration	Milisegundos	El tiempo que se tardó en completar el último punto de control	Aplicación	Esta métrica mide el tiempo que se tardó en completar el punto de control más reciente. Si el valor de esta métrica aumenta, esto puede indicar que hay un problema con la aplicación, como una pérdida de memoria o un cuello de botella. En algunos casos, puede solucionar este problema deshabilitando los puntos de control.

Métrica	Unidad	Descripción	Nivel	Notas de uso
managedMemoryUsed*	Bytes	La cantidad de memoria gestionada en uso actualmente.	Aplicación, Operador, Tarea, Paralelismo	<p>*Disponible para Managed Service para Apache Flink que ejecutan únicamente la versión 1.13 de Flink.</p> <p>Esto se refiere a la memoria gestionada por Flink fuera de la pila de Java. Se usa para el backend de estado de RockSDB y también está disponible para las aplicaciones.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
managedMemoryTotal*	Bytes	La cantidad total de memoria gestionada.	Aplicación, Operador, Tarea, Paralelismo	<p>*Disponible para Managed Service para Apache Flink que ejecutan únicamente la versión 1.13 de Flink.</p> <p>Esto se refiere a la memoria gestionada por Flink fuera de la pila de Java. Se usa para el backend de estado de RockSDB y también está disponible para las aplicaciones. La métrica ManagedMemoryUtilizations solo tiene en cuenta métricas de memoria específicas, como la memoria administrada (uso de memoria fuera de la JVM</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
				para procesos nativos como el backend de estado de RockSDB)
managedMemoryUtilization*	Porcentaje	Derivado por/managedMemoryUsed/managedMemoryTotal	Aplicación, Operador, Tarea, Paralelismo	<p>*Disponible para Managed Service para Apache Flink que ejecutan únicamente la versión 1.13 de Flink.</p> <p>Esto se refiere a la memoria gestionada por Flink fuera de la pila de Java. Se usa para el backend de estado de RockSDB y también está disponible para las aplicaciones.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
numberOfFailedCheckpoints	Recuento	La cantidad de veces que los puntos de control arrojaron error.	Aplicación	Puede utilizar esta métrica para supervisar el estado y el progreso de las aplicaciones. Los puntos de control pueden fallar debido a problemas con las aplicaciones, como problemas de rendimiento o permisos.

Métrica	Unidad	Descripción	Nivel	Notas de uso
numRecordsIn*	Recuento	La cantidad total de registros que ha recibido esta aplicación, operador o tarea.	Aplicación, Operador, Tarea, Paralelismo	<p>*Para aplicar la estadística SUM durante un período de tiempo (segundos/minuto):</p> <ul style="list-style-type: none"> • Seleccionar la métrica en el nivel correcto. Si está rastreando la métrica de un operador, debe seleccionar las métricas del operador correspondientes. • Como el servicio gestionado de Apache Flink toma 4 instantáneas de métricas por minuto, se debe utilizar la siguiente métrica matemática:

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>m1/4, donde m1 es la estadística SUM durante un período (segundo/ minuto)</p> <p>El nivel de la métrica especifica si esta métrica mide la cantidad total de registros que ha recibido toda la aplicación, un operador específico o una tarea específica.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
numRecordsInPeriod*	Recuento/ segundo	La cantidad total de registros por segundo que esta aplicación, operador o tarea ha recibido.	Aplicación, Operador, Tarea, Paralelismo	<p>*Para aplicar la estadística SUM durante un período de tiempo (segundos/minuto):</p> <ul style="list-style-type: none"> • Seleccione la métrica en el nivel correcto. Si está rastreando la métrica de un operador, debe seleccionar las métricas del operador correspondientes. • Como el servicio gestionado de Apache Flink toma 4 instantáneas de métricas por minuto, se debe utilizar la siguiente métrica matemática:

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>m1/4, donde m1 es la estadística SUM durante un período (segundo/ minuto)</p> <p>El nivel de la métrica especifica si esta métrica mide la cantidad total de registros que ha recibido toda la aplicación, un operador específico o una tarea específica por segundo.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
numRecordsOut*	Recuento	La cantidad total de registros que esta aplicación, operador o tarea ha recibido.	Aplicación, Operador, Tarea, Paralelismo	<p>*Para aplicar la estadística SUM durante un período de tiempo (segundos/minuto):</p> <ul style="list-style-type: none"> • Seleccion e la métrica en el nivel correcto. Si está rastreando la métrica de un operador, debe seleccionar las métricas del operador correspondientes. • Como el servicio gestionado de Apache Flink toma 4 instantáneas de métricas por minuto, se debe utilizar la siguiente métrica matemática:

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>m1/4, donde m1 es la estadística SUM durante un período (segundo/ minuto)</p> <p>El nivel de la métrica especifica si esta métrica mide la cantidad total de registros que toda la aplicación, un operador específico o una tarea específica ha emitido.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
numLateRecordsDropped*	Recuento	Aplicación, Operador, Tarea, Paralelismo		<p>*Para aplicar la estadística SUM durante un período de tiempo (segundos/minuto):</p> <ul style="list-style-type: none"> • Seleccionar la métrica en el nivel correcto. Si está rastreando la métrica de un operador, debe seleccionar las métricas del operador correspondientes. • Como el servicio gestionado de Apache Flink toma 4 instantáneas de métricas por minuto, se debe utilizar la siguiente métrica matemática:

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>$m1/4$, donde $m1$ es la estadística SUM durante un período (segundo/minuto)</p> <p>La cantidad de registros que este operador o tarea ha perdido por llegar tarde.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
numRecordsOutPerSecond*	Recuento/ segundo	La cantidad total de registros por segundo que esta aplicación, operador o tarea ha emitido.	Aplicación, Operador, Tarea, Paralelismo	<p>*Para aplicar la estadística SUM durante un período de tiempo (segundos/minuto):</p> <ul style="list-style-type: none"> • Seleccionar la métrica en el nivel correcto. Si está rastreando la métrica de un operador, debe seleccionar las métricas del operador correspondientes. • Como el servicio gestionado de Apache Flink toma 4 instantáneas de métricas por minuto, se debe utilizar la siguiente métrica matemática:

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>m1/4, donde m1 es la estadística SUM durante un período (segundo/ minuto)</p> <p>El nivel de la métrica especifica si esta métrica mide la cantidad total de registros que ha emitido toda la aplicación, un operador específico o una tarea específica por segundo.</p>

Métrica	Unidad	Descripción	Nivel	Notas de uso
oldGenerationGCCount	Recuento	La cantidad total de operaciones antiguas de recopilación de elementos no utilizados que se han llevado a cabo en todos los administradores de tareas.	Aplicación	
oldGenerationGCTime	Milisegundos	El tiempo total dedicado a realizar antiguas operaciones de recopilación de elementos no utilizados.	Aplicación	Puede usar esta métrica para monitorear la suma, el promedio y el tiempo máximo de recopilación de elementos no utilizados.
threadCount	Recuento	La cantidad total de subprocesos activos utilizados por la aplicación.	Aplicación	Esta métrica mide la cantidad de subprocesos utilizados por el código de la aplicación. No es lo mismo que el paralelismo de la aplicación.

Métrica	Unidad	Descripción	Nivel	Notas de uso
uptime	Milisegundos	El tiempo que el trabajo se ha estado ejecutando sin interrupción.	Aplicación	Puede usar esta métrica para determinar si un trabajo se está ejecutando correctamente. Esta métrica devuelve -1 para los trabajos completados.

Métricas del conector de flujo de datos de Kinesis

AWS emite todos los registros de flujo de datos de Kinesis además de los siguientes:

Métrica	Unidad	Descripción	Nivel	Notas de uso
millisbehindLatest	Milisegundos	La cantidad de milisegundos que el consumidor está detrás de la cabecera de la transmisión, lo que indica el retraso del consumidor con respecto a la hora actual.	Aplicación (para Stream), paralelismo (para) ShardId	<ul style="list-style-type: none"> Un valor de 0 indica que el procesamiento de registros está actualizado y que no hay nuevos registros para procesar en este momento. La métrica de una partición en particular se puede especificar

Métrica	Unidad	Descripción	Nivel	Notas de uso
				<p>mediante el nombre del flujo y el identificador de la partición.</p> <ul style="list-style-type: none"> • Un valor de -1 indica que el servicio aún no ha registrado o ningún valor para la métrica.
bytesRequestedPerFetch	Bytes	Los bytes solicitados a <code>getRecords</code> en una sola llamada.	Aplicación (para Stream), Paralelismo (para) <code>ShardId</code>	

Métricas de Amazon MSK Connector

AWS emite todos los registros de Amazon MSK además de los siguientes:

Métrica	Unidad	Descripción	Nivel	Notas de uso
currentOffsets	N/A	El desfase de lectura actual del consumidor, para cada partición. La métrica de una partición en particular se puede especificar	Aplicación (para el tema), Paralelismo (para) <code>PartitionId</code>	

Métrica	Unidad	Descripción	Nivel	Notas de uso
		ar mediante el nombre del tema y el identificador de la partición.		
commitsFailed	N/A	La cantidad total de errores de confirmación de desplazamientos de Kafka, si están activados la confirmación de desplazamientos y los puntos de control.	Aplicación, Operador, Tarea, Paralelismo	Confirmar los desplazamientos a Kafka solo es una forma de exponer el progreso de los consumidores, por lo que un error al confirmar los desplazamientos no afecta a la integridad de los desplazamientos divididos con puntos de control de Flink.
commitsSuccessful	N/A	La cantidad total de desplazamientos confirmados satisfactoriamente con Kafka, si la confirmación de desplazamientos y los puntos de control están activados.	Aplicación, Operador, Tarea, Paralelismo	

Métrica	Unidad	Descripción	Nivel	Notas de uso
committed offsets	N/A	Los últimos desplazamientos confirmados correctamente para Kafka, para cada partición . La métrica de una partición en particular se puede especificar mediante el nombre del tema y el identificador de la partición.	Aplicación (para el tema), Paralelismo (para) PartitionId	
records_lag_max	Recuento	El retraso máximo en términos de la cantidad de registros para cualquier partición de esta ventana	Aplicación, Operador, Tarea, Paralelismo	
bytes_consumed_rate	Bytes	Cantidad media de bytes consumidos por segundo para un tema	Aplicación, Operador, Tarea, Paralelismo	

Métricas de Apache Zeppelin

En el caso de los portátiles Studio, AWS emite las siguientes métricas a nivel de la aplicación: KPIs, cpuUtilization, heapMemoryUtilization, oldGenerationGCtime,

oldGenerationGCCount, y threadCount. Además, emite las métricas que se muestran en la siguiente tabla, también a nivel de la aplicación.

Métrica	Unidad	Descripción	Nombre de Prometheus
zeppelinCPUUtilization	Porcentaje	Porcentaje total de utilización de la CPU en el servidor Apache Zeppelin.	process_cpu_usage
zeppelinHeapMemoryUtilization	Porcentaje	Porcentaje general de utilización de la memoria dinámica en el servidor Apache Zeppelin.	jvm_memory_used_bytes
zeppelinThreadCount	Recuento	La cantidad total de subprocesos activos utilizados por el servidor Apache Zeppelin.	jvm_threads_live_threads
zeppelinWaitingJobs	Recuento	La cantidad de trabajos de Apache Zeppelin en cola esperando un subproceso.	jetty_threads_jobs
zeppelinServerUptime	Segundos	El tiempo total que el servidor ha estado en funcionamiento.	process_uptime_seconds

Visualización de métricas CloudWatch

Puede ver CloudWatch las métricas de su aplicación mediante la CloudWatch consola de Amazon o la AWS CLI.

Para ver las métricas mediante la CloudWatch consola

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En el panel de navegación, seleccione Métricas.
3. En el panel CloudWatch Métricas por categoría de Managed Service for Apache Flink, elija una categoría de métricas.
4. En el panel superior, desplácese para ver la lista completa de métricas.

Para ver métricas mediante la AWS CLI

- En el símbolo del sistema, ejecute el siguiente comando.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Configuración de los CloudWatch niveles de informes de métricas

Puede controlar el nivel de métricas de la aplicación que crea su aplicación. Managed Service para Apache Flink admite los siguientes niveles de métricas:

- **Aplicación:** la aplicación solo informa del nivel más alto de métricas para cada aplicación. De forma predeterminada, las métricas del servicio gestionado para Apache Flink se publican a nivel de la aplicación.
- **Tarea:** la aplicación informa sobre las dimensiones métricas específicas de la tarea para las métricas definidas con el nivel de generación de informes de métricas de la tarea, como el número de registros que entran y salen de la aplicación por segundo.
- **Operador:** la aplicación informa de las dimensiones métricas específicas del operador para las métricas definidas con el nivel de informes de métricas del operador, como las métricas de cada operación de filtro o mapa.
- **Paralelismo:** la aplicación informa métricas de nivel `Task` y `Operator` para cada subproceso de ejecución. El nivel de informe no se recomienda para aplicaciones con una configuración de paralelismo superior a 64 debido a los costos excesivos.

Note

Solo debe usar este nivel de métrica para solucionar problemas debido a la cantidad de datos de métricas que genera el servicio. Solo puede establecer este nivel de métrica mediante la CLI. Este nivel de métrica no está disponible en la consola.

El nivel predeterminado es Aplicación. La aplicación informa de las métricas en el nivel actual y en todos los niveles superiores. Por ejemplo, si el nivel de informes está establecido en Operador, la aplicación informa de las métricas de la aplicación, la tarea y el operador.

El nivel de informes de CloudWatch métricas se establece mediante el `MonitoringConfiguration` parámetro de la [CreateApplication](#) acción o el `MonitoringConfigurationUpdate` parámetro de la [UpdateApplication](#) acción. En el siguiente ejemplo de solicitud de la [UpdateApplication](#) acción, se establece el nivel de informe de CloudWatch métricas en Tarea:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "MonitoringConfigurationUpdate": {
        "ConfigurationTypeUpdate": "CUSTOM",
        "MetricsLevelUpdate": "TASK"
      }
    }
  }
}
```

También puede configurar el nivel de registro mediante el parámetro `LogLevel` de la acción [CreateApplication](#) o el parámetro `LogLevelUpdate` de la acción [UpdateApplication](#). Puede utilizar los siguientes niveles de registro:

- **ERROR**: registra los eventos de error potencialmente recuperables.
- **WARN**: registra los eventos de advertencia que pueden provocar un error.
- **INFO**: registra los eventos informativos.
- **DEBUG**: registra los eventos de depuración generales.

Para obtener más información sobre los niveles de registro de Log4j, consulte [Niveles de registro personalizados](#) en la documentación de [Apache Log4j](#).

Uso de métricas personalizadas con Amazon Managed Service para Apache Flink

Managed Service for Apache Flink expone 19 métricas CloudWatch, incluidas las métricas de uso y rendimiento de los recursos. Además, puede crear sus propias métricas para realizar un seguimiento de los datos específicos de la aplicación, como el procesamiento de eventos o el acceso a recursos externos.

Este tema contiene las siguientes secciones:

- [Cómo funciona](#)
- [Ejemplos](#)
- [Visualización de métricas personalizadas](#)

Cómo funciona

Las métricas personalizadas de Managed Service para Apache Flink utilizan el sistema de métricas de Apache Flink. Las métricas de Apache Flink tienen los siguientes atributos:

- Tipo: el tipo de métrica describe cómo mide e informa los datos. Los tipos de métricas de Apache Flink disponibles incluyen Recuento, Indicador, Histograma y Medidor. Para obtener más información sobre los tipos de métricas de Apache Flink, consulte [Tipos de métricas](#).

Note

AWS CloudWatch Metrics no admite el tipo de métrica Histograma Apache Flink. CloudWatch solo puede mostrar las métricas de Apache Flink de los tipos Count, Gauge y Meter.

- Alcance: el ámbito de una métrica consta de su identificador y un conjunto de pares clave-valor que indican cómo se informará a la métrica. CloudWatch El identificador de una métrica consta de los elementos siguientes:
 - El alcance del sistema, que indica el nivel en el que se informa de la métrica (por ejemplo, el operador).

- Un ámbito de usuario, que define atributos como las variables de usuario o los nombres de los grupos de métricas. Estos atributos se definen mediante [MetricGroup.addGroup\(key, value\)](#) o [MetricGroup.addGroup\(name\)](#).

Para obtener más información sobre esta métrica, consulte [Alcance](#).

Para obtener más información sobre las métricas de Apache Flink, consulte [Métricas](#) en la [documentación de Apache Flink](#).

Para crear una métrica personalizada en su servicio gestionado para Apache Flink, puede acceder al sistema de métricas de Apache Flink desde cualquier función de usuario que se amplíe mediante una llamada. RichFunction [GetMetricGroup](#) Este método devuelve un [MetricGroup](#) objeto que puede utilizar para crear y registrar métricas personalizadas. El servicio gestionado para Apache Flink informa de todas las métricas creadas con la clave KinesisAnalytics de grupo. CloudWatch Las métricas personalizadas que defina tienen las siguientes características:

- Su métrica personalizada tiene un nombre de métrica y un nombre de grupo. Estos nombres deben estar compuestos por caracteres alfanuméricos.
- Los atributos que defina en el ámbito del usuario (excepto el grupo de KinesisAnalytics métricas) se publican como CloudWatch dimensiones.
- Las métricas personalizadas se publican en el nivel Application de forma predeterminada.
- Las dimensiones (Tarea/Operador/Paralelismo) se añaden a la métrica en función del nivel de supervisión de la aplicación. El nivel de supervisión de la aplicación se establece mediante el [MonitoringConfiguration](#) parámetro de la [CreateApplication](#) acción o el [MonitoringConfigurationUpdate](#) parámetro o de la [UpdateApplication](#) acción.

Ejemplos

Los siguientes ejemplos de código muestran cómo crear una clase de mapeo que cree e incremente una métrica personalizada, y cómo implementar la clase de mapeo en su aplicación agregándola a un objeto DataStream.

Métrica personalizada de recuento de registros

El siguiente ejemplo de código muestra cómo crear una clase de mapeo que cree una métrica que cuente los registros de un flujo de datos (la misma funcionalidad que la métrica numRecordsIn):

```
private static class NoOpMapperFunction extends RichMapFunction<String, String> {
```

```

private transient int valueToExpose = 0;
private final String customMetricName;

public NoOpMapperFunction(final String customMetricName) {
    this.customMetricName = customMetricName;
}

@Override
public void open(Configuration config) {
    getRuntimeContext().getMetricGroup()
        .addGroup("KinesisAnalytics")
        .addGroup("Program", "RecordCountApplication")
        .addGroup("NoOpMapperFunction")
        .gauge(customMetricName, (Gauge<Integer>) () -> valueToExpose);
}

@Override
public String map(String value) throws Exception {
    valueToExpose++;
    return value;
}
}

```

En el ejemplo anterior, la variable `valueToExpose` se incrementa para cada registro que procesa la aplicación.

Tras definir la clase de mapeo, se crea una secuencia en la aplicación que implementa el mapa:

```

DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));

```

Para ver el código completo de esta aplicación, consulte [Métrica personalizada de recuento de registros](#).

Métrica personalizada de recuento de palabras

El siguiente ejemplo de código muestra cómo crear una clase de mapeo que cree una métrica que cuente palabras en un flujo de datos:

```

private static final class Tokenizer extends RichFlatMapFunction<String, Tuple2<String, Integer>> {

```

```
private transient Counter counter;

@Override
public void open(Configuration config) {
    this.counter = getRuntimeContext().getMetricGroup()
        .addGroup("KinesisAnalytics")
        .addGroup("Service", "WordCountApplication")
        .addGroup("Tokenizer")
        .counter("TotalWords");
}

@Override
public void flatMap(String value, Collector
```

En el ejemplo anterior, la variable `counter` se incrementa para cada palabra que procesa la aplicación.

Tras definir la clase de mapeo, se crea una secuencia en la aplicación que implementa el mapa:

```
// Split up the lines in pairs (2-tuples) containing: (word,1), and
// group by the tuple field "0" and sum up tuple field "1"
DataStream<Tuple2<String, Integer>> wordCountStream = input.flatMap(new
    Tokenizer()).keyBy(0).sum(1);

// Serialize the tuple to string format, and publish the output to kinesis sink
wordCountStream.map(tuple -> tuple.toString()).addSink(createSinkFromStaticConfig());
```

Para ver el código completo de esta aplicación, consulte [Métrica personalizada de recuento de palabras](#).

Visualización de métricas personalizadas

Las métricas personalizadas de la aplicación aparecen en la consola de CloudWatch métricas del AWS/KinesisAnalyticspanel de control, en el grupo de métricas de la aplicación.

Uso de CloudWatch alarmas con Amazon Managed Service para Apache Flink

Con las alarmas CloudWatch métricas de Amazon, observas una CloudWatch métrica durante un período de tiempo que especifiques. La alarma realiza una o varias acciones según el valor de la métrica o expresión con respecto a un umbral durante varios períodos de tiempo. Un ejemplo de acción es el envío de una enviar una notificación a un tema de Amazon Simple Notification Service (Amazon SNS).

Para obtener más información sobre CloudWatch las alarmas, consulte [Uso de Amazon CloudWatch Alarms](#).

Alarmas recomendadas

Esta sección contiene las alarmas recomendadas para supervisar el servicio gestionado para las aplicaciones de Apache Flink.

La tabla describe las alarmas recomendadas y tiene las siguientes columnas:

- **Expresión métrica:** la métrica o expresión métrica que se va a comprobar si se compara con el umbral.
- **Estadística:** la estadística que se utiliza para comprobar la métrica, por ejemplo, el promedio.
- **Umbral:** el uso de esta alarma requiere que determine un umbral que defina el límite del rendimiento esperado de la aplicación. Debe determinar este umbral supervisando la aplicación en condiciones normales.
- **Descripción:** causas que podrían activar esta alarma y posibles soluciones para esta afección.

Expresiones de métricas	Estadística	Threshold	Descripción
tiempo de inactividad > 0	Average	0	A downtime greater than zero indicates

Expresiones de métricas	Estadística	Threshold	Descripción
			<p>that the application has failed. If the value is larger than 0, the application is not processing any data. Recommended for all applications. The <code>Tiempo de inactividad</code> metric measures the duration of an outage. A downtime greater than zero indicates that the application has failed. For troubleshooting, see La aplicación se está reiniciando.</p>

Expresiones de métricas	Estadística	Threshold	Descripción
<code>RATE (numberOfFailedControlPoints) > 0</code>	Average	0	<p>This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be tolerable if checkpoints fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitoring <code>RATE(numberOfFailedCheckpoints)</code> to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpointing progress. The application saves state data to checkpoints when it's healthy. Checkpointing can fail due to timeouts if the application isn't making progress in</p>

Expresiones de métricas	Estadística	Threshold	Descripción
Operador. numRecord sOutPerSecond threshold	Average	The minimum number of records emitted from the application during normal conditions.	processing the input data. For troubleshooting, see Se está agotando el tiempo para llegar al punto de control. Recommended for all applications. Falling below this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see El rendimiento es demasiado lento.

Expresiones de métricas	Estadística	Threshold	Descripción
<code>records_lag_max millisbehindLatest > threshold</code>	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the application has already fallen behind. Recommended for all applications. Use the <code>records_lag_max</code> metric for a Kafka source, or the <code>millisbehindLatest</code> for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see

Expresiones de
métricas

Estadística

Threshold

Descripción

[El rendimiento es demasiado lento.](#)

Expresiones de métricas	Estadística	Threshold	Descripción
<code>lastCheckpointDuration > threshold</code>	Maximum	The maximum expected checkpoint duration during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also consider monitoring the change rate with <code>TARIFA (lastCheckpointSize)</code> and <code>TARIFA (lastCheckpointDuration)</code> . If the <code>lastCheckpointDuration</code> continuously increases, rising above this threshold can indicate that the application isn't making expected

Expresiones de métricas	Estadística	Threshold	Descripción
			progress on the input data, or that there are problems with application health such as backpressure. For troubleshooting, see Crecimiento de estado ilimitado .

Expresiones de métricas	Estadística	Threshold	Descripción
<code>lastCheckpointSize > threshold</code>	Maximum	The maximum expected checkpoint size during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also consider monitoring the change rate with <code>TARIFA (lastCheckpointSize)</code> and <code>TARIFA (lastCheckpointDuration)</code> . If the <code>lastCheckpointSize</code> continuously increases, rising above this threshold can indicate that the application is accumulating state

Expresiones de métricas	Estadística	Threshold	Descripción
heapMemoryUtilization > threshold	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected heapMemoryUtilization size during normal conditions, with a recommended value of 90 percent.	<p data-bbox="1187 252 1510 766">data. If the state data becomes too large, the application can run out of memory when recovering from a checkpoint, or recovering from a checkpoint might take too long. For troubleshooting, see Crecimiento de estado ilimitado.</p> <p data-bbox="1187 808 1510 1659">You can use this metric to monitor the maximum memory utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources. You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see Escalado.</p>

Expresiones de métricas	Estadística	Threshold	Descripción
<code>cpuUtilization > threshold</code>	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected <code>cpuUtilization</code> size during normal conditions, with a recommended value of 80 percent.	You can use this metric to monitor the maximum CPU utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources. You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see Escalado .
<code>threadsCount > threshold</code>	Maximum	The maximum expected <code>threadsCount</code> size during normal conditions.	You can use this metric to watch for thread leaks in task managers across the application. If this metric reaches this threshold, check your application code for threads being created without being closed.

Expresiones de métricas	Estadística	Threshold	Descripción
<code>(oldGarbageCollectionTime * 100) / 60_000</code> durante un período de 1 minuto') > threshold	Maximum	The maximum expected oldGarbageCollectionHora duration. We recommend setting a threshold such that typical garbage collection time is 60 percent of the specified threshold , but the correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
TASA (oldGarbageCollectionrecuento) > threshold	Maximum	The maximum expected oldGarbageCollectionContar under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
<code>Operator.currentOutputWatermark - Operator.currentInputWatermark</code> > threshold	Minimum	The minimum expected watermark increment under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that either the application is processing increasingly older events, or that an upstream subtask has not sent a watermark in an increasingly long time.

Escribir mensajes personalizados en los CloudWatch registros

Puede escribir mensajes personalizados en el registro de la aplicación Managed Service for Apache Flink. Para ello, utilice la biblioteca [log4j](#) o la biblioteca [Simple Logging Facade for Java \(SLF4J\)](#) de Apache.

Temas

- [Escriba en los CloudWatch registros mediante Log4J](#)
- [Escriba en los CloudWatch registros con SLF4J](#)

Escriba en los CloudWatch registros mediante Log4J

1. Agregue las siguientes dependencias al archivo `pom.xml` de su aplicación:

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.6.1</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.6.1</version>
</dependency>
```

2. Incluya el objeto de la biblioteca:

```
import org.apache.logging.log4j.Logger;
```

3. Cree una instancia del objeto `Logger`, pasando su clase de aplicación:

```
private static final Logger log =
  LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. Escriba en el registro usando `log.info`. Se escribe una gran cantidad de mensajes en el registro de la aplicación. Para facilitar el filtrado de los mensajes personalizados, utilice el nivel de registro `INFO` de la aplicación.

```
log.info("This message will be written to the application's CloudWatch log");
```

La aplicación escribe un registro en el registro con un mensaje similar al siguiente:

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

Escriba en los CloudWatch registros con SLF4J

1. Agregue la siguiente dependencia al archivo `pom.xml` de su aplicación:

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.7</version>
  <scope>runtime</scope>
</dependency>
```

2. Incluya los objetos de la biblioteca:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

3. Cree una instancia del objeto `Logger`, pasando su clase de aplicación:

```
private static final Logger log =
  LoggerFactory.getLogger(YourApplicationClass.class);
```

4. Escriba en el registro usando `log.info`. Se escribe una gran cantidad de mensajes en el registro de la aplicación. Para facilitar el filtrado de los mensajes personalizados, utilice el nivel de registro `INFO` de la aplicación.

```
log.info("This message will be written to the application's CloudWatch log");
```

La aplicación escribe un registro en el registro con un mensaje similar al siguiente:

```
{
  "locationInformation": "com.amazonaws.services.managed-
  flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticssus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

Registro de llamadas de API de Managed Service para Apache Flink con AWS CloudTrail

El servicio gestionado para Apache Flink está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio del servicio gestionado para Apache Flink. CloudTrail captura todas las llamadas a la API del servicio gestionado de Apache Flink como eventos. Las llamadas capturadas incluyen las llamadas realizadas desde el servicio administrado para la consola de Apache Flink y las llamadas de código al servicio administrado para las operaciones de la API de Apache Flink. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Managed Service for Apache Flink. 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 usted CloudTrail, puede determinar la solicitud que se realizó a Managed Service for Apache Flink, la dirección IP desde la que se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la [Guía del AWS CloudTrail usuario](#).

Servicio gestionado para Apache Flink Information en CloudTrail

CloudTrail está activado en su AWS cuenta al crear la cuenta. Cuando se produce una actividad en Managed Service for Apache Flink, 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 últimos eventos de la cuenta de AWS. Para obtener más información, consulte [Visualización de eventos con el historial de CloudTrail eventos](#).

Para mantener un registro continuo de los eventos en su cuenta de AWS, incluidos los eventos de Managed Service para Apache Flink, cree un registro de seguimiento. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De manera predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. 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 Integraciones y servicios 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 del servicio gestionado para Apache Flink se registran CloudTrail y se documentan en la referencia de la [API del servicio gestionado para Apache Flink](#). Por ejemplo, las llamadas a las [UpdateApplication](#) acciones [CreateApplication](#) y las acciones generan entradas en los archivos de CloudTrail registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario AWS Identity and Access Management (IAM) o credenciales de usuario raíz.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte el [Elemento userIdentity de CloudTrail](#).

Comprender las entradas de archivos de registro de Managed Service para Apache Flink

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted 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 muestra las [DescribeApplication](#) acciones [AddApplicationCloudWatchLoggingOption](#).

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::012345678910:user/Alice",
        "accountId": "012345678910",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
      },
      "eventTime": "2019-03-07T01:19:47Z",
      "eventSource": "kinesisanalytics.amazonaws.com",
      "eventName": "AddApplicationCloudWatchLoggingOption",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
      "requestParameters": {
        "applicationName": "cloudtrail-test",
        "currentApplicationVersionId": 1,
        "cloudWatchLoggingOption": {
          "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
        }
      },
      "responseElements": {
        "cloudWatchLoggingOptionDescriptions": [
          {
```

```

        "cloudWatchLoggingOptionId": "2.1",
        "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
    }
  ],
  "applicationVersionId": 2,
  "applicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678910:application/cloudtrail-test"
},
"requestID": "18dfb315-4077-11e9-afd3-67f7af21e34f",
"eventID": "d3c9e467-db1d-4cab-a628-c21258385124",
"eventType": "AwsApiCall",
"apiVersion": "2018-05-23",
"recipientAccountId": "012345678910"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2019-03-12T02:40:48Z",
  "eventSource": "kinesisanalytics.amazonaws.com",
  "eventName": "DescribeApplication",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
  "requestParameters": {
    "applicationName": "sample-app"
  },
  "responseElements": null,
  "requestID": "3e82dc3e-4470-11e9-9d01-e789c4e9a3ca",
  "eventID": "90ffe8e4-9e47-48c9-84e1-4f2d427d98a5",
  "eventType": "AwsApiCall",
  "apiVersion": "2018-05-23",
  "recipientAccountId": "012345678910"
}
]
}

```

Afinar el desempeño en Amazon Managed Service para Apache Flink

Este tema describe técnicas para monitorear y mejorar el desempeño de su aplicación Managed Service para Apache Flink.

Temas

- [Solución de problemas de desempeño](#)
- [Prácticas recomendadas de desempeño](#)
- [Monitoreo del desempeño](#)

Solución de problemas de desempeño

Esta sección contiene una lista de síntomas que puede comprobar para diagnosticar y solucionar problemas de desempeño.

Si el origen de datos es una transmisión de Kinesis, los problemas de desempeño suelen presentarse como una métrica `millisbehindLatest` alta o creciente. En el caso de otros orígenes, puede comprobar una métrica similar que represente el retraso en la lectura del origen.

La ruta de datos

Cuando investigue un problema de desempeño con su aplicación, tenga en cuenta todo el recorrido que siguen sus datos. Los siguientes componentes de la aplicación pueden convertirse en cuellos de botella en el desempeño y crear retrasos si no se diseñan u obtienen adecuadamente:

- Orígenes y destinos de datos: asegúrese de que los recursos externos con los que interactúa su aplicación sean correctamente obtenidos para el rendimiento que experimentará la aplicación.
- Datos de estado: asegúrese de que su aplicación no interactúe con el almacén estatal con demasiada frecuencia.

Puede optimizar el serializador que utiliza su aplicación. El serializador Kryo predeterminado puede manejar cualquier tipo serializable, pero puede usar un serializador más eficiente si su aplicación solo almacena datos en tipos POJO. Para obtener información sobre los serializadores de Apache Flink, consulte [Tipos de datos y serialización](#) en la [documentación de Apache Flink](#).

- **Operadores:** asegúrese de que la lógica empresarial implementada por sus operadores no sea demasiado complicada o de que no cree o utilice recursos en cada registro procesado. Asegúrese también de que su aplicación no cree ventanas deslizantes o de salto de tamaño constante con demasiada frecuencia.

Solución de problemas de desempeño

Esta sección contiene posibles soluciones a los problemas de desempeño.

Temas

- [Niveles de monitoreo de CloudWatch](#)
- [Métricas de CPU de la aplicación](#)
- [Paralelismo de aplicaciones](#)
- [Registro de la aplicación](#)
- [Paralelismo del operador](#)
- [Lógica de la aplicación](#)
- [Memoria de aplicaciones](#)

Niveles de monitoreo de CloudWatch

Compruebe que los niveles de monitoreo de CloudWatch no estén configurados de forma demasiado detallada.

La configuración del nivel de registro de monitoreo de Debug genera una gran cantidad de tráfico, lo que puede generar retrasos. Solo debe usarlo mientras investiga activamente los problemas de la aplicación.

Si su aplicación tiene una configuración de `Parallelism` alta, el uso del nivel de métricas de monitoreo de `Parallelism` también generará una gran cantidad de tráfico que puede provocar retrasos. Utilice este nivel de métricas únicamente cuando `Parallelism` sea bajo para su aplicación o cuando investigue problemas con la aplicación.

Para obtener más información, consulte [Niveles de monitoreo de aplicaciones](#).

Métricas de CPU de la aplicación

Comprueba la métrica CPU de la aplicación. Si esta métrica supera el 75 por ciento, puede permitir que la aplicación se asigne más recursos al habilitar el escalado automático.

Si el escalado automático está habilitado, la aplicación asigna más recursos si el uso de la CPU es superior al 75 por ciento durante 15 minutos. Para obtener más información acerca del escalado, consulte la sección [Administrar correctamente el escalado](#) y la [Escalado](#) a continuación.

Note

Una aplicación solo escalará automáticamente en respuesta al uso de la CPU. La aplicación no escalará automáticamente en respuesta a otras métricas del sistema, como `heapMemoryUtilization`. Si su aplicación tiene un alto nivel de uso de otras métricas, aumente el paralelismo de la aplicación manualmente.

Paralelismo de aplicaciones

Aumente el paralelismo de la aplicación. El paralelismo de la aplicación se actualiza mediante el parámetro `ParallelismConfigurationUpdate` de la acción [UpdateApplication](#).

El número máximo de KPIs de una aplicación es de 64 de forma predeterminada y se puede aumentar solicitando un aumento del límite.

También es importante asignar el paralelismo a cada operador en función de su carga de trabajo, en lugar de simplemente aumentar el paralelismo de la aplicación. Consulte [Paralelismo del operador](#) a continuación.

Registro de la aplicación

Compruebe si la aplicación registra una entrada para cada registro que se esté procesando. Escribir una entrada de registro para cada registro en momentos en que la aplicación tenga un alto rendimiento provocará graves embotellamientos en el procesamiento de datos. Para comprobar si existe esta condición, busque en los registros las entradas de registro que la aplicación escriba con cada registro que procese. Para obtener más información sobre cómo leer registros de aplicación, consulte [the section called “Análisis de registros”](#).

Paralelismo del operador

Compruebe que la carga de trabajo de su aplicación se distribuya uniformemente entre los procesos de trabajo.

Para obtener información sobre cómo ajustar la carga de trabajo de los operadores de su aplicación, consulte [Escalado de operadores](#).

Lógica de la aplicación

Examine la lógica de la aplicación para detectar operaciones ineficientes o que no funcionen, como acceder a una dependencia externa (como una base de datos o un servicio web), acceder al estado de la aplicación, etc. Una dependencia externa también puede obstaculizar el desempeño si no funciona o no se puede acceder a ella de forma fiable, lo que puede provocar que la dependencia externa devuelva errores HTTP 500.

Si su aplicación utiliza una dependencia externa para enriquecer o procesar de otro modo los datos entrantes, considere la posibilidad de utilizar una ES asíncrona en su lugar. Para obtener más información, consulte [E/S asíncrona](#) en la [documentación de Apache Flink](#).

Memoria de aplicaciones

Compruebe si su aplicación presenta fugas de recursos. Si su aplicación no elimina correctamente los subprocesos o la memoria, es posible que observe que las métricas `millisBehindLatest`, `CheckpointSize` y `CheckpointDuration` aumentan de golpe o gradualmente. Esta condición también puede provocar fallos en el administrador de tareas o en el administrador de funciones.

Prácticas recomendadas de desempeño

En esta sección se describen las consideraciones especiales a la hora de diseñar una aplicación orientada al desempeño.

Administrar correctamente el escalado

Esta sección contiene información sobre la administración del escalado a nivel de aplicación y a nivel de operador.

Esta sección contiene los siguientes temas:

- [Gestionar correctamente el escalado de las aplicaciones](#)

- [Gestionar correctamente el escalado del operador](#)

Gestionar correctamente el escalado de las aplicaciones

Puede utilizar el escalado automático para gestionar los picos inesperados en la actividad de las aplicaciones. Las KPU de su aplicación se incrementarán automáticamente si se cumplen los siguientes criterios:

- El ajuste de escala automático está activado para la aplicación.
- El uso de la CPU permanece por encima del 75 por ciento durante 15 minutos.

Si el escalado automático está activado, pero el uso de la CPU no se mantiene en este umbral, la aplicación no escalará las KPU. Si experimenta un aumento en el uso de la CPU que no alcanza este umbral o un aumento en una métrica de uso diferente, como por ejemplo `heapMemoryUtilization`, aumente el escalado manualmente para que su aplicación pueda gestionar los picos de actividad.

Note

Si la aplicación ha agregado automáticamente más recursos mediante el escalado automático, la aplicación liberará los nuevos recursos tras un período de inactividad. La reducción de los recursos afectará temporalmente al desempeño.

Para obtener más información acerca del escalado, consulte [Escalado](#).

Gestionar correctamente el escalado del operador

Puede mejorar el desempeño de su aplicación verificando que la carga de trabajo de la aplicación se distribuya de manera uniforme entre los procesos de trabajo y que los operadores de la aplicación dispongan de los recursos del sistema que necesitan para mantener la estabilidad y el desempeño.

Puede establecer el paralelismo para cada operador en el código de su aplicación mediante el parámetro `parallelism`. Si no establece el paralelismo para un operador, éste utilizará el ajuste de paralelismo a nivel de aplicación. Los operadores que utilizan la configuración de paralelismo a nivel de aplicación pueden utilizar todos los recursos del sistema disponibles para la aplicación, lo que hace que la aplicación se vuelva inestable.

Para determinar mejor el paralelismo de cada operador, tenga en cuenta los requisitos de recursos relativos del operador en comparación con los demás operadores de la aplicación. Defina los operadores que consumen más recursos con una configuración de paralelismo de operadores más alta que los operadores que consumen menos recursos.

El paralelismo total de los operadores de la aplicación es la suma del paralelismo de todos los operadores de la aplicación. Para ajustar el paralelismo total de los operadores de la aplicación, debe determinar la mejor relación entre este y el total de tareas disponibles para la aplicación. Una relación estable típica entre el paralelismo total del operador y los intervalos de tareas es de 4:1, es decir, la aplicación tiene un intervalo de tareas disponible por cada cuatro subtareas del operador disponibles. Una aplicación con operadores que consumen más recursos puede necesitar una relación de 3:1 o 2:1, mientras que una aplicación con operadores que consumen menos recursos puede ser estable con una relación de 10:1.

Puede establecer la relación que utilizará el operador usando [Propiedades de tiempo de ejecución](#), para ajustar el paralelismo del operador sin necesidad de compilar ni cargar el código de la aplicación.

El siguiente ejemplo de código muestra cómo establecer el paralelismo de los operadores como una relación ajustable del paralelismo de la aplicación actual:

```
Map<String, Properties> applicationProperties =
    KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

    applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

Para obtener información sobre las subtareas, las tareas disponibles y otros recursos de la aplicación, consulte [Recursos de aplicaciones](#).

Para controlar la distribución de la carga de trabajo entre los procesos de trabajo de la aplicación, utilice la configuración `Parallelism` y el método de partición `KeyBy`. Para obtener más información, consulte los siguientes temas en la [documentación de Apache Flink](#):

- [Ejecución en paralelo](#)
- [Transformaciones de `DataStream`](#)

Supervisar el uso de los recursos de dependencia externa

Si hay un cuello de botella en el desempeño en un destino (como Kinesis Streams, Kinesis Data Firehose, DynamoDB u OpenSearch Service), la aplicación experimentará retrasos. Compruebe que las dependencias externas se hayan obtenido correctamente para el rendimiento de su aplicación.

Note

Los errores en otros servicios pueden provocar errores en la aplicación. Si observa errores en su aplicación, compruebe los registros de CloudWatch de sus servicios de destino para ver si hay errores.

Ejecute la aplicación Apache Flink localmente

Para solucionar problemas de memoria, puede ejecutar la aplicación en una instalación local de Flink. Esto le permitirá acceder a herramientas de depuración, como el rastreo de pilas y los volcados de pilas, que no están disponibles al ejecutar la aplicación en Managed Service for Apache Flink.

Para obtener información sobre cómo crear una instalación local de Flink, consulte el [tutorial de configuración local](#) en la [documentación de Apache Flink](#).

Monitoreo del desempeño

En esta sección se describen las herramientas para monitorear el desempeño de una aplicación.

Monitoreo de desempeño con métricas de CloudWatch

Se supervisa el uso de los recursos, el rendimiento, los puntos de control y el tiempo de inactividad de su aplicación mediante las métricas de CloudWatch. Para obtener información sobre el uso de registros de CloudWatch con su aplicación Managed Service para Apache Flink, consulte [Métricas y dimensiones en Managed Service para Apache Flink](#).

Supervisión del desempeño mediante registros CloudWatch y Alarmas Alarms

Se supervisan las condiciones de error que podrían provocar problemas de desempeño mediante registros CloudWatch.

Las condiciones de error aparecen en las entradas de registro cuando el estado de la tarea de Apache Flink cambia del estado RUNNING al estado FAILED.

Las alarmas de CloudWatch se utilizan para crear notificaciones sobre problemas de desempeño, como el uso de los recursos o las métricas de puntos de control que superan un umbral seguro, o cambios inesperados en el estado de las aplicaciones.

Para obtener información sobre cómo crear alarmas de CloudWatch para una aplicación de Managed Service para Apache Flink, consulte [Alarmas](#).

Cuota de Managed Service para Apache Flink y Studio Notebook

Al trabajar con Amazon Managed Service para Apache Flink, tenga en cuenta la siguiente cuota:

- Puede crear hasta 50 aplicaciones de Managed Service para Apache Flink por región en su cuenta. Puede crear un caso para solicitar en aplicaciones adicionales a través del formulario de incremento de cuota de servicio. Para obtener más información, consulte el [Centro de AWS Support](#).

Para obtener una lista de las regiones que admiten Managed Service para Apache Flink, consulte [Regiones y puntos de conexión de Managed Service para Apache Flink](#).

- El número de unidades de procesamiento de Kinesis (KPU) está limitado a 64 por defecto. Para obtener instrucciones sobre cómo solicitar un aumento de esta cuota, consulte Solicitar un aumento de cuota en [Service Quotas](#). Asegúrese de especificar el prefijo de la aplicación a la que se debe aplicar el nuevo límite de la KPU.

Con Managed Service para Apache Flink, a su cuenta de AWS se le cobran los recursos asignados, en lugar de los que utiliza su aplicación. A usted se le cobra una tarifa por hora en función del número máximo de KPU que se utilizan para ejecutar su aplicación de procesamiento de flujos. Una sola KPU le proporciona 1 vCPU y 4 GB de memoria. Para cada KPU, el servicio también proporciona 50 GiB de almacenamiento de aplicaciones en ejecución.

- Puede crear hasta 1000 [Instantáneas](#) de Managed Service para Apache Flink por aplicación.
- Puede asignar hasta 50 etiquetas por aplicación.
- El tamaño máximo de un archivo JAR de aplicación es de 512 MiB. Si supera esta cuota, su aplicación no se iniciará.

Para los cuadernos Studio, se aplican las siguientes cuotas. Para solicitar cuotas más altas, [cree un caso de asistencia](#).

- `websocketMessageSize` = 5 MiB
- `noteSize` = 5 MiB
- `noteCount` = 1000
- Max cumulative UDF size = 100 MiB
- Max cumulative dependency jar size = 300 MiB

Mantenimiento de Managed Service para Apache Flink

Managed Service para Apache Flink parchea sus aplicaciones periódicamente con actualizaciones de seguridad del sistema operativo y de la imagen del contenedor para mantener el cumplimiento y cumplir los objetivos de seguridad de AWS. En la siguiente tabla se muestra el intervalo de tiempo predeterminado durante el cual Managed Service para Apache Flink realiza este tipo de mantenimiento. El mantenimiento de su aplicación puede realizarse en cualquier momento durante el período de tiempo correspondiente a su región. Es posible que su aplicación experimente un tiempo de inactividad de 10 a 30 segundos durante este proceso de mantenimiento. Sin embargo, la duración real del tiempo de inactividad depende del estado de la aplicación. Para obtener información sobre cómo minimizar el impacto de este tiempo de inactividad, consulte [the section called “Tolerancia a fallos: puntos de control y puntos de almacenamiento”](#).

Para cambiar el intervalo de tiempo durante el cual Managed Service para Apache Flink realiza el mantenimiento de su aplicación, utilice la API [UpdateApplicationMaintenanceConfiguration](#).

Región	Periodo de mantenimiento
AWS GovCloud (Oeste de EE. UU.)	06:00 — 14:00 UTC
AWS GovCloud (Este de EE. UU.)	03:00 — 11:00 UTC
Este de EE. UU. (Norte de Virginia)	03:00 — 11:00 UTC
Este de EE. UU. (Ohio)	03:00 — 11:00 UTC
Oeste de EE. UU. (Norte de California)	06:00 — 14:00 UTC
Oeste de EE. UU. (Oregón)	06:00 — 14:00 UTC
Asia-Pacífico (Hong Kong)	13:00 — 21:00 UTC
Asia-Pacífico (Bombay)	16:30 — 00:30 UTC
Asia-Pacífico (Hyderabad)	16:30 — 00:30 UTC
Asia-Pacífico (Seúl)	13:00 — 21:00 UTC
Asia-Pacífico (Singapur)	14:00 — 22:00 UTC

Región	Periodo de mantenimiento
Asia-Pacífico (Sídney)	12:00 — 20:00 UTC
Asia-Pacífico (Yakarta)	15:00 — 23:00 UTC
Asia-Pacífico (Tokio)	13:00 — 21:00 UTC
Canadá (centro)	03:00 — 11:00 UTC
China (Pekín)	13:00 — 21:00 UTC
China (Ningxia)	13:00 — 21:00 UTC
Europa (Fráncfort)	06:00 — 14:00 UTC
Europa (Zúrich)	20:00 — 04:00 UTC
Europa (Irlanda)	22:00 — 06:00 UTC
Europa (Londres)	22:00 — 06:00 UTC
Europa (Estocolmo)	23:00 — 07:00 UTC
Europa (Milán)	21:00 — 05:00 UTC
Europa (España)	21:00 — 05:00 UTC
África (Ciudad del Cabo)	20:00 — 04:00 UTC
Europa (Irlanda)	22:00 — 06:00 UTC
Europa (Londres)	23:00 — 07:00 UTC
Europa (París)	23:00 — 07:00 UTC
Europa (Estocolmo)	23:00 — 07:00 UTC
Medio Oriente (Baréin)	13:00 — 21:00 UTC
Medio Oriente (EAU)	18:00 — 02:00 UTC

Región	Periodo de mantenimiento
América del Sur (São Paulo)	19:00 — 03:00 UTC
Israel (Tel Aviv)	20:00 — 04:00 UTC

Establezca un UUID para todos los operadores

Cuando Managed Service para Apache Flink inicia un trabajo de Flink para una aplicación con una instantánea, es posible que el trabajo de Flink no se inicie debido a ciertos problemas. Uno de ellos es la disparidad de ID de los operadores. Flink espera identificadores de operador explícitos y consistentes para los operadores de gráficos de trabajos de Flink. Si no se establece de forma explícita, Flink genera automáticamente un ID para los operadores. Esto se debe a que Flink usa estos ID de operador para identificar de forma única a los operadores en un gráfico de trabajos y los usa para almacenar el estado de cada operador en un punto de guardado.

El problema de disparidad de ID de los operadores se produce cuando Flink no encuentra una correlación de 1:1 entre los ID de los operadores de un gráfico de trabajos y los ID de los operadores definidos en un punto de guardado. Esto ocurre cuando no se establecen ID de operador explícitos y consistentes, y Flink genera automáticamente ID de operador que pueden no ser consistentes con la creación de cada gráfico de trabajo. La probabilidad de que las aplicaciones presenten este problema es alta durante las operaciones de mantenimiento. Para evitarlo, recomendamos a los clientes configurar el UUID para todos los operadores en código flink. Para obtener más información, consulte el tema [Establecer un UUID para todos los operadores](#) en [Preparación para producción](#).

Preparación para la producción

Esta es una recopilación de aspectos importantes de la ejecución de aplicaciones de producción en Managed Service para Apache Flink. No se trata de una lista exhaustiva, sino de lo mínimo a lo que hay que prestar atención antes de poner una aplicación en producción.

Aplicaciones de pruebas de carga

Algunos problemas de las aplicaciones solo se manifiestan cuando hay mucha carga. Hemos visto casos en los que las aplicaciones parecían estar en buen estado, pero un incidente operativo incrementó considerablemente la carga de trabajo de la aplicación. Esto puede ocurrir de forma totalmente independiente de la propia aplicación: si el origen de datos o el receptor de datos no están disponibles durante un par de horas, la aplicación Flink no podrá avanzar. Una vez que se soluciona el problema, hay una acumulación de datos sin procesar que puede agotar por completo los recursos disponibles. La carga puede entonces amplificar errores o problemas de rendimiento que no habían surgido antes.

Por lo tanto, es esencial realizar las pruebas de carga adecuadas para las aplicaciones de producción. Las preguntas que deben responderse durante esas pruebas de carga incluyen:

- ¿La aplicación es estable bajo una carga elevada sostenida?
- ¿Puede la aplicación seguir ocupando un punto de almacenamiento en momentos de máxima carga?
- ¿Cuánto tiempo demora procesar un acumulado de 1 hora? ¿Y cuánto tiempo demora para 24 horas (en función de la retención máxima de los datos en la transmisión)?
- ¿Aumenta el rendimiento de la aplicación cuando se escala la aplicación?

Cuando se consume desde un flujo de datos, estos escenarios se pueden simular produciendo en el flujo durante algún tiempo. A continuación, inicie la aplicación y haga que consuma datos desde el comienzo del tiempo. Por ejemplo, utilice una posición de inicio de TRIM_HORIZON en el caso de un Flujo de Datos Kinesis.

Máximo paralelismo

El paralelismo máximo define el paralelismo máximo al que se puede escalar una aplicación con estado. Esto se define cuando se crea el estado por primera vez, y no hay forma de escalar el operador más allá de este máximo sin descartar el estado.

El paralelismo máximo se establece cuando se crea el estado por primera vez.

De forma predeterminada, el paralelismo máximo está definido como:

- 128, si el paralelismo ≤ 128
- $\text{MIN}(\text{nextPowerOfTwo}(\text{parallelism} + (\text{parallelism} / 2)), 2^{15})$: si el paralelismo > 128

Si planea escalar su aplicación a un paralelismo superior a 128 grados, debe definir explícitamente el paralelismo máximo.

El paralelismo máximo se puede definir a nivel de aplicación, con `env.setMaxParallelism(x)` o con un solo operador. A menos que se especifique lo contrario, todos los operadores heredan el paralelismo máximo de la aplicación.

Para obtener más información, consulte [Establecer un paralelismo máximo explícito](#) en la documentación de Flink.

Establezca un UUID para todos los operadores

Se utiliza un UUID en la operación en la que Flink asigna un punto de almacenamiento a un operador individual. Si se establece un UUID específico para cada operador, se obtiene un mapeo estable para el proceso de restauración del punto de almacenamiento.

```
.map(...).uid("my-map-function")
```

Para obtener más información, consulte la [Lista de verificación de preparación para la producción](#).

Prácticas recomendadas de Managed Service para Apache Flink

Esta sección contiene información y recomendaciones para desarrollar un servicio gestionado estable y eficaz para las aplicaciones de Apache Flink.

Temas

- [Tolerancia a fallos: puntos de control y puntos de almacenamiento](#)
- [Versiones de conector no compatibles](#)
- [Rendimiento y paralelismo](#)
- [Establecer el paralelismo por operador](#)
- [Registro](#)
- [Codificación](#)
- [Administración de credenciales](#)
- [Lectura de fuentes con pocas particiones](#)
- [Intervalo de actualización del bloc de notas de Studio](#)
- [Rendimiento óptimo del bloc de notas de Studio](#)
- [Cómo afectan las estrategias de marcas de agua y las particiones inactivas a las ventanas temporales](#)
- [Establezca un UUID para todos los operadores](#)
- [Añádale ServiceResourceTransformer al complemento Maven Shade](#)

Tolerancia a fallos: puntos de control y puntos de almacenamiento

Utilice puntos de control y puntos de almacenamiento para implementar la tolerancia a errores en su aplicación de Managed Service para Apache Flink. Tenga en cuenta las siguientes consideraciones al desarrollar y mantener la aplicación:

- Le recomendamos que deje habilitada la verificación de puntos de control de la aplicación. Los puntos de control permiten que la aplicación tolere los errores durante el mantenimiento programado, así como en caso de que se produzcan fallos inesperados debidos a problemas de servicio, fallos de dependencia de la aplicación y otros problemas. Para obtener más información sobre mantenimiento, consulte [Mantenimiento](#).

- Establezca [ApplicationSnapshotConfiguration:: SnapshotsEnabled](#) como `false` durante el desarrollo de la aplicación o la solución de problemas. Se crea una instantánea cada vez que se detiene una aplicación, lo que puede provocar problemas si la aplicación se encuentra en mal estado o no funciona correctamente. Establecer `SnapshotsEnabled` en `true` cuando la aplicación esté en producción y esté estable.

Note

Se recomienda que la aplicación cree una instantánea varias veces al día para que se reinicie correctamente con los datos de estado correctos. La frecuencia correcta de las instantáneas depende de la lógica de negocios de la aplicación. Tomar instantáneas frecuentes le permite recuperar datos más recientes, pero aumenta el costo y requiere más recursos del sistema.

Para obtener información sobre la supervisión del tiempo de inactividad de las aplicaciones, consulte [Métricas y dimensiones en Managed Service para Apache Flink](#).

Para obtener más información acerca de la implementación de tolerancias ante fallos, consulte [Tolerancia a errores](#).

Versiones de conector no compatibles

La versión 1.15 de Managed Service para Apache Flink impedirá automáticamente que las aplicaciones se inicien o se actualicen si utilizan versiones de Kinesis Connector no compatibles (incluidas en los JAR de las aplicaciones). Al actualizar a la versión 1.15 de Managed Service para Apache Flink, asegúrese de utilizar el Kinesis Connector más reciente. Se trata de cualquier versión igual o posterior a la 1.15.2. Las versiones restantes no serán compatibles con Managed Service para Apache Flink, ya que pueden provocar problemas de coherencia o fallos, ya que la característica Parada con punto de guardado impide realizar operaciones de parada o actualización sin problemas.

Rendimiento y paralelismo

Su aplicación puede escalarse para cumplir con cualquier nivel de rendimiento ajustando el paralelismo de la aplicación y evitando los problemas de rendimiento. Tenga en cuenta las siguientes consideraciones al desarrollar y mantener la aplicación:

- Compruebe que todas las fuentes y receptores de las aplicaciones estén suficientemente provisionadas y que no estén siendo restringidas. Si las fuentes y los receptores son otros AWS servicios, supervise el uso [CloudWatch](#) de esos servicios.
- En el caso de aplicaciones con un paralelismo muy alto, compruebe si los niveles altos de paralelismo se aplican a todos los operadores de la aplicación. De forma predeterminada, Apache Flink aplica el mismo paralelismo de aplicación a todos los operadores del gráfico de la aplicación. Esto puede provocar problemas de aprovisionamiento en las fuentes o los receptores, o provocar cuellos de botella en el procesamiento de los datos de los operadores. Puede cambiar el paralelismo de cada operador en el código con [setParallelism](#).
- Comprenda el significado de la configuración de paralelismo para los operadores de su aplicación. Si cambia el paralelismo de un operador, es posible que no pueda restaurar la aplicación a partir de una instantánea creada cuando el operador tenía un paralelismo que no es compatible con la configuración actual. Para obtener más información sobre cómo configurar el paralelismo del operador, consulte [Establecer el paralelismo máximo para los operadores de forma explícita](#).

Para obtener más información acerca de la implementación del escalado, consulte [Escalado](#).

Establecer el paralelismo por operador

De forma predeterminada, todos los operadores tienen el paralelismo establecido en el nivel de la aplicación. Puede anular el paralelismo de un solo operador utilizando la API mediante `DataStream.setParallelism(x)`. Puede establecer el paralelismo de un operador en cualquier paralelismo igual o inferior al paralelismo de la aplicación.

Si es posible, defina el paralelismo del operador como una función del paralelismo de la aplicación. De esta forma, el paralelismo del operador variará con el paralelismo de la aplicación. Si utiliza el escalado automático, por ejemplo, todos los operadores variarán su paralelismo en la misma proporción:

```
int appParallelism = env.getParallelism();  
...  
...ops.setParallelism(appParallelism/2);
```

En algunos casos, es posible que desee establecer el paralelismo del operador en una constante. Por ejemplo, establecer el paralelismo de una fuente de Kinesis Stream con la cantidad de particiones. En estos casos, debería considerar la posibilidad de incluir el paralelismo del operador

como parámetro de configuración de la aplicación para poder cambiarlo sin cambiar el código, si necesita, por ejemplo, volver a fragmentar el flujo de origen.

Registro

Puede supervisar el rendimiento y las condiciones de error de su aplicación mediante los registros. CloudWatch Tenga en cuenta las siguientes consideraciones al configurar los registros de la aplicación:

- Habilita el CloudWatch registro de la aplicación para poder depurar cualquier problema de tiempo de ejecución.
- No cree una entrada de registro para cada registro que se esté procesando en la aplicación. Esto provoca graves cuellos de botella durante el procesamiento y puede provocar retrasos en el procesamiento de los datos.
- Cree CloudWatch alarmas que le notifiquen cuando la aplicación no se ejecute correctamente. Para obtener más información, consulte [Alarmas](#)

Para obtener más información acerca de la implementación de los registros, consulte [Registro y supervisión](#).

Codificación

Puede hacer que su aplicación funcione y sea estable mediante las prácticas de programación recomendadas. Tenga en cuenta las siguientes consideraciones al escribir el código de aplicación:

- No utilice `system.exit()` en el código de la aplicación, ni en el método `main` de la aplicación ni en las funciones definidas por el usuario. Si quiere cerrar la aplicación desde el código, lance una excepción derivada de `Exception` o `RuntimeException` que contenga un mensaje sobre el problema de la aplicación.

Tenga en cuenta lo siguiente sobre cómo gestiona el servicio esta excepción:

- Si la excepción proviene del método `main` de la solicitud, el servicio la incluirá en una `ProgramInvocationException` cuando la solicitud pase al estado `RUNNING` y el administrador del trabajo no podrá enviar el trabajo.
- Si la excepción proviene de una función definida por el usuario, el administrador de tareas fallará en la tarea y la reiniciará, y los detalles de la excepción se escribirán en el registro de excepciones.

- Considere la posibilidad de sombrear el archivo JAR de la aplicación y las dependencias incluidas en él. Se recomienda sombrear cuando haya posibles conflictos en los nombres de los paquetes entre la aplicación y el tiempo de ejecución de Apache Flink. Si se produce un conflicto, es posible que los registros de la aplicación contengan alguna excepción de tipo `java.util.concurrent.ExecutionException`. Para obtener más información sobre cómo sombrear el archivo JAR de la aplicación, consulte [Apache Maven Shade Plugin](#).

Administración de credenciales

No debe incluir credenciales a largo plazo en las aplicaciones de producción (ni en ninguna otra). Es probable que las credenciales a largo plazo se registren en un sistema de control de versiones y se pierdan fácilmente. En su lugar, puede asociar una función a la aplicación Managed Service para Apache Flink y conceder privilegios a esa función. La aplicación Flink en ejecución puede recoger credenciales temporales del entorno con los privilegios correspondientes. En caso de que sea necesaria la autenticación para un servicio que no esté integrado de forma nativa con IAM, por ejemplo, una base de datos que requiera un nombre de usuario y una contraseña para la autenticación, debería considerar la posibilidad de almacenar los secretos en [AWS Secrets Manager](#).

Muchos servicios nativos de AWS admiten la autenticación:

- [Kinesis Data StreamsProcessTaxiStream](#): `.java`
- Amazon MSK: <https://github.com/aws/aws-msk-iam-auth-using-the-amazon-msk/#-library-for-iam-authentication>
- [Amazon Elasticsearch Service](#): `.java AmazonElasticsearchSink`
- Amazon S3: funciona de forma inmediata en un servicio gestionado para Apache Flink

Lectura de fuentes con pocas particiones

Al leer desde Apache Kafka o un flujo de datos de Kinesis, es posible que no coincida el paralelismo del flujo (es decir, el número de particiones de Kafka y el número de particiones de Kinesis) y el paralelismo de la aplicación. Con un diseño ingenuo, el paralelismo de una aplicación no puede ampliarse más allá del paralelismo de un flujo: cada subtarea de un operador fuente solo puede leer de 1 o más particiones. Esto significa que, en el caso de una transmisión con solo 2 fragmentos y una aplicación con un paralelismo de 8, solo se consumen realmente dos subtareas de la transmisión y 6 subtareas permanecen inactivas. Esto puede limitar considerablemente el

rendimiento de la aplicación, especialmente si la deserialización es cara y la lleva a cabo la fuente (que es la opción predeterminada).

Para mitigar este efecto, puedes escalar la transmisión. Pero eso no siempre es deseable o posible. Alternativamente, puede reestructurar la fuente para que no realice ninguna serialización y simplemente transmita la `byte[]`. A continuación, puede [volver a equilibrar](#) los datos para distribuirlos uniformemente entre todas las tareas y, a continuación, deserializar los datos allí. De esta forma, puede aprovechar todas las subtareas para la deserialización y esta operación, potencialmente costosa, ya no está limitada por el número de particiones de la transmisión.

Intervalo de actualización del bloc de notas de Studio

Si cambia el intervalo de actualización de resultados de párrafos, configúrelo en un valor de al menos 1000 milisegundos.

Rendimiento óptimo del bloc de notas de Studio

Probamos con la siguiente afirmación y obtuvimos el mejor rendimiento cuando `events-per-second` multiplicado por `number-of-keys` dio menos de 25 000 000. Esto fue para `events-per-second` por debajo de 150 000.

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

Cómo afectan las estrategias de marcas de agua y las particiones inactivas a las ventanas temporales

Al leer los eventos de Apache Kafka y flujo de datos de Kinesis, la fuente puede establecer la hora del evento en función de los atributos de la transmisión. En el caso de Kinesis, la hora del evento es igual a la hora aproximada de llegada de los eventos. Sin embargo, establecer la hora del evento en el origen de los eventos no es suficiente para que una aplicación de Flink utilice la hora del evento. El origen también debe generar marcas de agua que propaguen la información sobre la hora del evento desde el origen a todos los demás operadores. La [documentación de Flink](#) ofrece una buena visión general de cómo funciona ese proceso.

De forma predeterminada, la marca de tiempo de un evento leído en Kinesis se establece en la hora de llegada aproximada determinada por Kinesis. Un requisito previo adicional para que la hora del evento funcione en la aplicación es una estrategia de marca de agua.

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

Luego, la estrategia de marca de agua se aplica a `DataStream` con el método `assignTimestampsAndWatermarks`. Hay algunas estrategias integradas útiles:

- `forMonotonousTimestamps()` solo utilizará la hora del evento (hora aproximada de llegada) y emitirá periódicamente el valor máximo como marca de agua (para cada subtarea específica)
- `forBoundedOutOfOrderness(Duration.ofSeconds(...))` similar a la estrategia anterior, pero utilizará el tiempo y la duración del evento para generar marcas de agua.

Esto funciona, pero hay un par de advertencias que debe tener en cuenta. Las marcas de agua se generan a nivel de subtarea y fluyen a través del gráfico del operador.

De la documentación de [Flink](#):

Cada subtarea paralela de una función fuente suele generar sus marcas de agua de forma independiente. Estas marcas de agua definen la hora del evento en esa fuente paralela en particular.

A medida que las marcas de agua recorren el programa de streaming, adelantan la hora del evento en los operadores a los que llegan. Cada vez que un operador adelanta la hora de su evento, genera una nueva marca de agua para sus operadores subsiguientes aguas abajo.

Algunos operadores consumen varios flujos de entrada; una unión, por ejemplo, o los operadores que siguen una función `KeyBy(...)` o `partition(...)`. El tiempo de evento actual de un operador de este tipo es el mínimo de los tiempos de evento de sus flujos de entrada. A medida que sus flujos de entrada actualizan los tiempos de sus eventos, también lo hace el operador.

Esto significa que, si una subtarea de origen consume contenido de una partición inactiva, los operadores intermedios no reciben nuevas marcas de agua de esa subtarea y, por lo tanto, el procesamiento se detiene para todos los operadores intermedios que utilizan ventanas temporales. Para evitarlo, los clientes pueden añadir la opción `withIdleness` a la estrategia de marcas de agua. Con esta opción, el operador excluye las marcas de agua de las subtareas previas inactivas al calcular la hora del evento del operador. Por lo tanto, las subtareas inactivas ya no bloquean el avance de la hora del evento en los operadores intermedios.

Sin embargo, la opción de inactividad con las estrategias de marca de agua integradas no adelantará la hora del evento si ninguna subtarea lee ningún evento, es decir, si no hay ningún evento en la

transmisión. Esto resulta especialmente visible en los casos de prueba en los que se lee un conjunto finito de eventos de la secuencia. Como el tiempo del evento no avanza después de haber leído el último evento, la última ventana (que contiene el último evento) nunca se cerrará.

Resumen

- el parámetro `withIdleness` no generará nuevas marcas de agua en caso de que una partición esté inactiva, sino que simplemente excluirá la última marca de agua enviada por las subtareas inactivas del cálculo de la marca de agua mínima en los operadores aguas abajo
- con las estrategias de marcas de agua integradas, la última ventana abierta nunca se cerrará (a menos que se envíen nuevos eventos que hagan avanzar la marca de agua, pero que creen una nueva ventana que luego permanecerá abierta)
- incluso si la transmisión de Kinesis establece la hora, pueden producirse eventos que lleguen tarde si una partición se consume más rápido que otras (por ejemplo, durante la inicialización de la aplicación o cuando se usa, `TRIM_HORIZON` cuando todas las particiones existentes se consumen en paralelo sin tener en cuenta su relación padre/hijo)
- el parámetro `withIdleness` de la estrategia de marca de agua parece dejar obsoleta la configuración específica de origen de Kinesis para las particiones inactivas (`ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`)

Ejemplo

La siguiente aplicación lee una transmisión y crea ventanas de sesión en función de la hora del evento.

```
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "eu-west-1");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "TRIM_HORIZON");

FlinkKinesisConsumer<String> consumer = new FlinkKinesisConsumer<>("...", new
    SimpleStringSchema(), consumerConfig);

WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(15));

env.addSource(consumer)
    .assignTimestampsAndWatermarks(s)
```

```

.map(new MapFunction<String, Long>() {
    @Override
    public Long map(String s) throws Exception {
        return Long.parseLong(s);
    }
})
.keyBy(1 -> 01)
.window(EventTimeSessionWindows.withGap(Time.seconds(10)))
.process(new ProcessWindowFunction<Long, Object, Long, TimeWindow>() {
    @Override
    public void process(Long aLong, ProcessWindowFunction<Long, Object, Long,
TimeWindow>.Context context, Iterable<Long>iterable, Collector<Object> collector)
throws Exception {
        long count = StreamSupport.stream(iterable.spliterator(), false).count();
        long timestamp = context.currentWatermark();

        System.out.print("XXXXXXXXXXXXXXXX Window with " + count + " events");
        System.out.println("; Watermark: " + timestamp + ", " +
Instant.ofEpochMilli(timestamp));

        for (Long l : iterable) {
            System.out.println(l);
        }
    }
});

```

En el siguiente ejemplo, se escriben 8 eventos en una secuencia de 16 particiones (los 2 primeros y el último evento ocurren en la misma partición).

```

$ aws kinesis put-record --stream-name hp-16 --partition-key 1 --data MQ==
$ aws kinesis put-record --stream-name hp-16 --partition-key 2 --data Mg==
$ aws kinesis put-record --stream-name hp-16 --partition-key 3 --data Mw==
$ date

{
  "ShardId": "shardId-000000000012",
  "SequenceNumber": "49627894338614655560500811028721934184977530127978070210"
}
{
  "ShardId": "shardId-000000000012",
  "SequenceNumber": "49627894338614655560500811028795678659974022576354623682"
}

```

```
{
  "ShardId": "shardId-000000000014",
  "SequenceNumber": "49627894338659257050897872275134360684221592378842022114"
}
Wed Mar 23 11:19:57 CET 2022

$ sleep 10
$ aws kinesisms put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kinesisms put-record --stream-name hp-16 --partition-key 5 --data NQ==
$ date

{
  "ShardId": "shardId-000000000010",
  "SequenceNumber": "49627894338570054070103749783042116732419934393936642210"
}
{
  "ShardId": "shardId-000000000014",
  "SequenceNumber": "49627894338659257050897872275659034489934342334017700066"
}
Wed Mar 23 11:20:10 CET 2022

$ sleep 10
$ aws kinesisms put-record --stream-name hp-16 --partition-key 6 --data Ng==
$ date

{
  "ShardId": "shardId-000000000001",
  "SequenceNumber": "49627894338369347363316974173886988345467035365375213586"
}
Wed Mar 23 11:20:22 CET 2022

$ sleep 10
$ aws kinesisms put-record --stream-name hp-16 --partition-key 7 --data Nw==
$ date

{
  "ShardId": "shardId-000000000008",
  "SequenceNumber": "49627894338525452579706688535878947299195189349725503618"
}
Wed Mar 23 11:20:34 CET 2022

$ sleep 60
$ aws kinesisms put-record --stream-name hp-16 --partition-key 8 --data OA==
$ date
```



```
{
  "ShardId": "shardId-000000000012",
  "SequenceNumber": "49627894338614655560500811029600823255837371928900796610"
}
```

Wed Mar 23 11:21:27 CET 2022

Esta entrada debería dar como resultado 5 ventanas de sesión: evento 1,2,3; evento 4,5; evento 6; evento 7; evento 8. Sin embargo, el programa solo muestra las 4 primeras ventanas.

```
11:59:21,529 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 5 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
127605887595351923798765477786913079296,EndingHashKey:
148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
49627894338480851089309627289524549239292625588395704418,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 5 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
127605887595351923798765477786913079296,EndingHashKey:
148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
49627894338480851089309627289524549239292625588395704418,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,530 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
148873535527910577765226390751398592512,EndingHashKey:
170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
49627894338503151834508157912666084957565273949901684850,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
212676479325586539664609129644855132160,EndingHashKey:
233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
49627894338570054070103749782090692112383219034419626146,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
```

```
148873535527910577765226390751398592512,EndingHashKey:
170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
49627894338503151834508157912666084957565273949901684850,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,531 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 4 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
106338239662793269832304564822427566080,EndingHashKey:
127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
49627894338458550344111096666383013521019977226889723986,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 4 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
106338239662793269832304564822427566080,EndingHashKey:
127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
49627894338458550344111096666383013521019977226889723986,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
85070591730234615865843651857942052864,EndingHashKey:
106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
49627894338436249598912566043241477802747328865383743554,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
63802943797675961899382738893456539648,EndingHashKey:
85070591730234615865843651857942052863},SequenceNumberRange: {StartingSequenceNumber:
49627894338413948853714035420099942084474680503877763122,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
319014718988379809496913694467282698240,EndingHashKey:
340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
49627894338681557796096402897798370703746460841949528306,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
297747071055821155530452781502797185024,EndingHashKey:
```

```
319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
49627894338659257050897872274656834985473812480443547874,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
85070591730234615865843651857942052864,EndingHashKey:
106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
49627894338436249598912566043241477802747328865383743554,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
63802943797675961899382738893456539648,EndingHashKey:
85070591730234615865843651857942052863},SequenceNumberRange: {StartingSequenceNumber:
49627894338413948853714035420099942084474680503877763122,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
21267647932558653966460912964485513216,EndingHashKey:
42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
49627894338369347363316974173816870647929383780865802258,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
191408831393027885698148216680369618944,EndingHashKey:
212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
49627894338547753324905219158949156394110570672913645714,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
255211775190703847597530955573826158592,EndingHashKey:
276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
```

```
49627894338614655560500811028373763548928515757431587010,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
170141183460469231731687303715884105728,EndingHashKey:
191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
49627894338525452579706688535807620675837922311407665282,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
21267647932558653966460912964485513216,EndingHashKey:
42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
49627894338369347363316974173816870647929383780865802258,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
233944127258145193631070042609340645376,EndingHashKey:
255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
49627894338592354815302280405232227830655867395925606578,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
42535295865117307932921825928971026432,EndingHashKey:
63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
49627894338391648108515504796958406366202032142371782690,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
276479423123262501563991868538311671808,EndingHashKey:
297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
```

```
49627894338636956305699341651515299267201164118937567442,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
42535295865117307932921825928971026432,EndingHashKey:
63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
49627894338391648108515504796958406366202032142371782690,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:23,209 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
191408831393027885698148216680369618944,EndingHashKey:
212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
49627894338547753324905219158949156394110570672913645714,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
212676479325586539664609129644855132160,EndingHashKey:
233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
49627894338570054070103749782090692112383219034419626146,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
event: 6; timestamp: 1648030822428, 2022-03-23T10:20:22.428Z
11:59:23,377 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
319014718988379809496913694467282698240,EndingHashKey:
340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
49627894338681557796096402897798370703746460841949528306,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,405 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
297747071055821155530452781502797185024,EndingHashKey:
319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
49627894338659257050897872274656834985473812480443547874,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
```

```

11:59:23,581 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
170141183460469231731687303715884105728,EndingHashKey:
191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
49627894338525452579706688535807620675837922311407665282,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,586 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
276479423123262501563991868538311671808,EndingHashKey:
297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
49627894338636956305699341651515299267201164118937567442,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:24,790 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
255211775190703847597530955573826158592,EndingHashKey:
276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
49627894338614655560500811028373763548928515757431587010,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 4; timestamp: 1648030809282, 2022-03-23T10:20:09.282Z
event: 3; timestamp: 1648030797697, 2022-03-23T10:19:57.697Z
event: 5; timestamp: 1648030810871, 2022-03-23T10:20:10.871Z
11:59:24,907 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
233944127258145193631070042609340645376,EndingHashKey:
255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
49627894338592354815302280405232227830655867395925606578,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 7; timestamp: 1648030834105, 2022-03-23T10:20:34.105Z
event: 1; timestamp: 1648030794441, 2022-03-23T10:19:54.441Z
event: 2; timestamp: 1648030796122, 2022-03-23T10:19:56.122Z
event: 8; timestamp: 1648030887171, 2022-03-23T10:21:27.171Z
XXXXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z

```

```
4
5
XXXXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
6
XXXXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030887170, 2022-03-23T10:21:27.170Z
7
```

La salida solo muestra 4 ventanas (falta la última ventana que contiene el evento 8). Esto se debe a la hora del evento y a la estrategia de marca de agua. La última ventana no se puede cerrar porque, con las estrategias de marca de agua predefinidas, el tiempo nunca pasa más allá de la hora del último evento que se lee desde la transmisión. Sin embargo, para que la ventana se cierre, el tiempo debe avanzar más de 10 segundos después del último evento. En este caso, la última marca de agua es 2022-03-23T 10:21:27.170Z, pero para que la ventana de la sesión se cierre, es necesaria una marca de agua 10 segundos y 1 ms después.

Si se elimina la opción `withIdleness` de la estrategia de marca de agua, no se cerrará nunca ninguna ventana de sesión, ya que la "marca de agua global" del operador de la ventana no podrá avanzar.

Tenga en cuenta que cuando se inicia la aplicación Flink (o si los datos están sesgados), es posible que algunos fragmentos se consuman más rápido que otros. Esto puede provocar que algunas marcas de agua se emitan demasiado pronto desde una subtarea (es posible que la subtarea emita la marca de agua en función del contenido de una partición sin haber consumido contenido de las demás particiones a las que está suscrita). Las formas de mitigarlo son distintas estrategias de marca de agua que añadan un margen de seguridad (`forBoundedOutOfOrderness(Duration.ofSeconds(30))`) o permitan explícitamente que los eventos (`allowedLateness(Time.minutes(5))`) lleguen tarde.

Establezca un UUID para todos los operadores

Cuando Managed Service para Apache Flink inicia un trabajo de Flink para una aplicación con una instantánea, es posible que el trabajo de Flink no se inicie debido a ciertos problemas. Uno de ellos es la disparidad de ID de los operadores. Flink espera identificadores de operador explícitos y consistentes para los operadores de gráficos de trabajos de Flink. Si no se establece de forma explícita, Flink genera automáticamente un ID para los operadores. Esto se debe a que Flink usa estos ID de operador para identificar de forma única a los operadores en un gráfico de trabajos y los usa para almacenar el estado de cada operador en un punto de guardado.

El problema de disparidad de ID de los operadores se produce cuando Flink no encuentra una correlación de 1:1 entre los ID de los operadores de un gráfico de trabajos y los ID de los operadores definidos en un punto de guardado. Esto ocurre cuando no se establecen ID de operador explícitos y consistentes, y Flink genera automáticamente ID de operador que pueden no ser consistentes con la creación de cada gráfico de trabajo. La probabilidad de que las aplicaciones presenten este problema es alta durante las operaciones de mantenimiento. Para evitarlo, recomendamos a los clientes configurar el UUID para todos los operadores en código flink. Para obtener más información, consulte el tema Establecer un UUID para todos los operadores en [Preparación para producción](#).

Añádalo ServiceResourceTransformer al complemento Maven Shade

Flink utiliza las [interfaces de proveedor de servicios \(SPI\)](#) de Java para cargar componentes como conectores y formatos. El uso de SPI en varias dependencias de Flink [puede provocar conflictos en el archivo uber-jar y comportamientos inesperados de las aplicaciones](#). Se recomienda añadir el plugin Maven [ServiceResourceTransformer](#)Shade, definido en el archivo pom.xml

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <executions>
        <execution>
          <id>shade</id>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <transformers combine.children="append">
              <!-- The service transformer is needed to merge META-
INF/services files -->
              <transformer
implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/>
              <!-- ... -->
            </transformers>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```



```
</plugin>
```

Stateful Functions de Apache Flink

[Stateful Functions](#) es una API que simplifica la creación de aplicaciones con estado distribuidas. Se basa en funciones con un estado persistente que pueden interactuar de forma dinámica con sólidas garantías de coherencia.

Una aplicación Stateful Functions es básicamente una aplicación de Apache Flink y, por lo tanto, se puede implementar en Managed Service para Apache Flink. Sin embargo, hay un par de diferencias entre el empaquetado de Stateful Functions para un clúster de Kubernetes y el de Managed Service para Apache Flink. El aspecto más importante de una aplicación Stateful Functions es que la [configuración del módulo](#) contiene toda la información de tiempo de ejecución necesaria para configurar el tiempo de ejecución de Stateful Functions. Por lo general, esta configuración se empaqueta en un contenedor específico de Stateful Functions y se implementa en Kubernetes. Pero eso no es posible con Managed Service para Apache Flink.

A continuación se presenta una adaptación del ejemplo de StateFun Python para Managed Service para Apache Flink:

Plantilla de aplicación de Apache Flink

En lugar de utilizar un contenedor de cliente para el tiempo de ejecución de Stateful Functions, los clientes pueden compilar un contenedor de aplicaciones de Flink que simplemente invoque el tiempo de ejecución de Stateful Functions y contenga las dependencias necesarias. En el caso de Flink 1.13, las dependencias requeridas son similares a las siguientes:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>statefun-flink-distribution</artifactId>
  <version>3.1.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
    </exclusion>
    <exclusion>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
    </exclusion>
  </exclusions>
```

```
</dependency>
```

Y el método principal de la aplicación Flink para invocar el tiempo de ejecución de Stateful Function es el siguiente:

```
public static void main(String[] args) throws Exception {
    final StreamExecutionEnvironment env =
        StreamExecutionEnvironment.getExecutionEnvironment();

    StatefulFunctionsConfig stateFunConfig = StatefulFunctionsConfig.fromEnvironment(env);

    stateFunConfig.setProvider((StatefulFunctionsUniverseProvider) (classLoader,
        statefulFunctionsConfig) -> {
        Modules modules = Modules.loadFromClassPath();
        return modules.createStatefulFunctionsUniverse(stateFunConfig);
    });

    StatefulFunctionsJob.main(env, stateFunConfig);
}
```

Tenga en cuenta que estos componentes son genéricos e independientes de la lógica que se implementa en Stateful Function.

Ubicación de la configuración del módulo

La configuración del módulo Stateful Functions debe incluirse en la ruta de la clase para que pueda detectarse en el tiempo de ejecución de Stateful Functions. Es mejor incluirlo en la carpeta de recursos de la aplicación Flink y empaquetarlo en el archivo jar.

De forma similar a una aplicación común de Apache Flink, puede usar Maven para crear un archivo uber jar e implementarlo en Managed Service para Apache Flink.

Información de versiones anteriores de Managed Service para Apache Flink

Este tema contiene información sobre el uso de Managed Service para Apache Flink con versiones anteriores de Apache Flink. Las versiones de Apache Flink compatibles con Managed Service para Apache Flink son la 1.15.2 (recomendada), 1.13.2, 1.11.1, 1.8.2 y 1.6.2.

Le recomendamos utilizar la última versión compatible de Apache Flink con su aplicación Managed Service para Apache Flink. La versión 1.15.2 de Apache Flink tiene las siguientes características:

- Compatibilidad con la [API de tablas de Apache Flink y SQL](#)
- Compatibilidad con aplicaciones Python.
- Compatibilidad con la versión 11 de Java y cualquier versión de Scala
- Un modelo de memoria mejorado
- Optimizaciones de RocksDB para aumentar la estabilidad de las aplicaciones
- Compatibilidad con el administrador de tareas y seguimiento de pilas en el panel de Apache Flink.

Este tema contiene las siguientes secciones:

- [Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink](#)
- [Creación de aplicaciones con Apache Flink 1.8.2](#)
- [Creación de aplicaciones con Apache Flink 1.6.2](#)
- [Actualización de aplicaciones](#)
- [Conectores disponibles en Apache Flink 1.6.2 y 1.8.2](#)
- [Introducción: Flink 1.13.2](#)
- [Introducción: Flink 1.11.1](#)
- [Introducción: Flink 1.8.2](#)
- [Introducción: Flink 1.6.2](#)

Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink

El conector Apache Flink Kinesis Streams no estaba incluido en Apache Flink antes de la versión 1.11. Para que su aplicación utilice el conector Apache Flink Kinesis con versiones anteriores de Apache Flink, debe descargar, compilar e instalar la versión de Apache Flink que utilice su aplicación. Este conector se utiliza para consumir datos de un flujo de Kinesis utilizado como fuente de aplicación o para escribir datos en un flujo de Kinesis utilizado para la salida de la aplicación.

Note

Asegúrese de que está compilando el conector con la [versión 0.14.0 o superior de KPL](#).

Para descargar e instalar el código fuente de la versión 1.8.2 de Apache Flink, haga lo siguiente:

1. Asegúrese de tener instalado [Apache Maven](#) y de que su variable de entorno `JAVA_HOME` apunte a un JDK en lugar de un JRE. Puede probar la instalación de Apache Maven con el siguiente comando:

```
mvn -version
```

2. Descargue el código fuente de la versión 1.8.2 de Apache Flink:

```
wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
```

3. Descomprima el código fuente de Apache Flink:

```
tar -xvf flink-1.8.2-src.tgz
```

4. Cambie al directorio del código fuente de Apache Flink:

```
cd flink-1.8.2
```

5. Compile e instale Apache Flink:

```
mvn clean install -Pinclude-kinesis -DskipTests
```

Note

Si está compilando Flink en Microsoft Windows, debe agregar el parámetro - `Drat.skip=true`.

Creación de aplicaciones con Apache Flink 1.8.2

Esta sección contiene información sobre los componentes que se utilizan para crear aplicaciones de Managed Service para Apache Flink que funcionan con Apache Flink 1.8.2.

Utilice las siguientes versiones de los componentes para las aplicaciones de Managed Service para Apache Flink:

Componente	Versión
Java	1.8 (recomendado)
Apache Flink	1.8.2
Managed Service para Apache Flink para el tiempo de ejecución de Flink (aws-kinesisanalytics-runtime)	1.0.1
Conectores de Flink de Managed Service para Apache Flink (aws-kinesisanalytics-runtime)	1.0.1
Apache Maven	3.1

Para compilar una aplicación con Apache Flink 1.8.2, ejecute Maven con el siguiente parámetro:

```
mvn package -Dflink.version=1.8.2
```

Para ver un ejemplo de un archivo `pom.xml` de una aplicación de Managed Service para Apache Flink que utiliza la versión 1.8.2 de Apache Flink, consulte la [aplicación de introducción de Managed Service para Apache Flink 1.8.2](#).

Para obtener información sobre cómo crear y usar el código de aplicación para una aplicación de Managed Service para Apache Flink, consulte [Creación de aplicaciones](#).

Creación de aplicaciones con Apache Flink 1.6.2

Esta sección contiene información sobre los componentes que se utilizan para crear aplicaciones de Managed Service para Apache Flink que funcionan con Apache Flink 1.6.2.

Utilice las siguientes versiones de los componentes para las aplicaciones de Managed Service para Apache Flink:

Componente	Versión
Java	1.8 (recomendado)
AWS Java SDK	1.11.379
Apache Flink	1.6.2
Managed Service para Apache Flink para el tiempo de ejecución de Flink (aws-kinesisanalytics-runtime)	1.0.1
Conectores de Flink de Managed Service para Apache Flink (aws-kinesisanalytics-runtime)	1.0.1
Apache Maven	3.1
Apache Beam	No compatible con Apache Flink 1.6.2

Note

Al utilizar la versión 1.0.1 del tiempo de ejecución de Managed Service para Apache Flink, debe especificar la versión de Apache Flink del archivo `pom.xml` en lugar de utilizar el parámetro `-Dflink.version` al compilar el código de la aplicación.

Para ver un ejemplo de un archivo `pom.xml` de una aplicación de Managed Service para Apache Flink que utiliza la versión 1.6.2 de Apache Flink, consulte la [aplicación de introducción de Managed Service para Apache Flink 1.6.2](#).

Para obtener información sobre cómo crear y usar el código de aplicación para una aplicación de Managed Service para Apache Flink, consulte [Creación de aplicaciones](#).

Actualización de aplicaciones

Para actualizar la versión de una aplicación de Managed Service para Apache Flink, debe actualizar el código de la aplicación, eliminar la aplicación anterior y crear una nueva aplicación con el código actualizado. Para ello, siga estos pasos:

- Cambie las versiones del tiempo de ejecución de Managed Service para Apache Flink y de los conectores de Flink de Managed Service para Apache Flink (`aws-kinesisanalytics-flink`) del archivo `pom.xml` de su aplicación a la versión 1.1.0.
- Elimine la propiedad `flink.version` del archivo `pom.xml` de su aplicación. Deberá proporcionar este parámetro al compilar el código de la aplicación en el siguiente paso.
- Vuelva a compilar el código de la aplicación usando el siguiente comando:

```
mvn package -Dflink.version=1.15.3
```

- Elimine la aplicación existente. Vuelva a crear la aplicación y elija Apache Flink versión 1.15.2 (versión recomendada) para el tiempo de ejecución de la aplicación.

Note

No puede usar instantáneas de las versiones anteriores de la aplicación.

Conectores disponibles en Apache Flink 1.6.2 y 1.8.2

El marco de Apache Flink contiene conectores para acceder a los datos desde una variedad de fuentes.

- Para obtener información sobre los conectores disponibles en el marco Apache Flink 1.6.2, consulte [Conectores \(1.6.2\)](#) en la documentación de [Apache Flink \(1.6.2\)](#).

- Para obtener información sobre los conectores disponibles en el marco Apache Flink 1.8.2, consulte [Conectores \(1.8.2\)](#) en la [documentación de Apache Flink \(1.8.2\)](#).

Introducción: Flink 1.13.2

En esta sección, se presentan los conceptos fundamentales del servicio gestionado para Apache Flink y la DataStream API. Describe las opciones disponibles para crear y probar sus aplicaciones. También proporciona instrucciones para instalar las herramientas necesarias para completar los tutoriales de esta guía y crear su primera aplicación.

Temas

- [Componentes de una aplicación de Managed Service para Apache Flink](#)
- [Requisitos previos para completar los ejercicios](#)
- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso siguiente](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)
- [Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)
- [Paso 4: Eliminar recursos de AWS](#)
- [Paso 5: Sigüientes pasos](#)

Componentes de una aplicación de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Java/Apache Maven o Scala que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

La aplicación de Managed Service para Apache Flink tiene los siguientes componentes:

- Propiedades de tiempo de ejecución: puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- Fuente: la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un bucket de Amazon S3, etc. Para obtener más información, consulte [Orígenes](#).

- **Operadores:** la aplicación procesa los datos mediante uno o más operadores. Un operador puede transformar, enriquecer o agregar datos. Para obtener más información, consulte [Operadores de la API de DataStream](#).
- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores](#).

Después de crear, compilar y empaquetar el código de la aplicación, debe cargar el paquete del código a un bucket de Amazon Simple Storage Service (Amazon S3). Luego debe crear la aplicación de Managed Service para Apache Flink. Introduzca la ubicación del paquete del código, un flujo de datos de Kinesis como origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos para completar los ejercicios

Para completar los pasos de esta guía, debe disponer de lo siguiente:

- [Java Development Kit \(JDK\), versión 11](#). Establezca la variable de entorno JAVA_HOME para señalar la ubicación de la instalación del JDK.
- Le recomendamos utilizar un entorno de desarrollo (como [Eclipse Java Neon](#) o [IntelliJ Idea](#)) para desarrollar y compilar su aplicación.
- [Cliente Git](#). Si aún no lo ha hecho, instale el cliente Git.
- [Apache Maven Compiler Plugin](#). Maven debe estar en su ruta de trabajo. Para probar la instalación de Apache Maven, introduzca lo siguiente:

```
$ mvn -version
```

Para empezar, vaya a [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#).

Paso 1: Configurar una cuenta de AWS y crear un usuario administrador

Registro para obtener una Cuenta de AWS

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

Cómo registrarse para una Cuenta de AWS

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

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

Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, [asigne acceso administrativo a un usuario administrativo](#) y utilice únicamente el usuario raíz para realizar [tareas que requieran acceso de usuario raíz](#).

AWS le enviará un correo electrónico de confirmación luego de completar el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en <https://aws.amazon.com/> y seleccionando Mi cuenta.

Crear un usuario administrativo

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

Protección de su Usuario raíz de la cuenta de AWS

1. Inicie sesión en la [AWS Management Console](#) como propietario de cuenta, elija Usuario raíz e ingrese el email de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Signing in as the root user](#) en la Guía del usuario de AWS Sign-In.

2. Active la autenticación multifactor (MFA) para el usuario raíz.

Para obtener instrucciones, consulte [Habilitar un dispositivo MFA virtual para el usuario raíz de la Cuenta de AWS \(consola\)](#) en la Guía del usuario de IAM.

Creación de un usuario administrativo

1. Activar IAM Identity Center

Para conocer las instrucciones, consulte [Habilitar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center.

2. En IAM Identity Center, otorga acceso administrativo a un usuario administrativo.

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

Cómo iniciar sesión como usuario administrativo

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

Para obtener ayuda para iniciar sesión con un usuario del IAM Identity Center, consulte [Iniciar sesión en el portal de acceso de AWS](#) en la Guía del Usuario de AWS Sign-In.

Conceder acceso programático

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

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

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • En AWS CLI, consulte Configuración de la AWS CLI para usar AWS IAM Identity Center en la Guía del usuario de AWS Command Line Interface.

¿Qué usuario necesita acceso programático?	Para	Mediante
		<ul style="list-style-type: none">• En los SDK de AWS, las herramientas y las API de AWS, consulte Autenticación de IAM Identity Center en la Guía de referencia de SDK de AWS y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del Usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para los SDK de AWS y las herramientas, consulte Autenticación mediante credenciales a largo plazo en la Guía de referencia de SDK de AWS y herramientas. • Para las API de AWS, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Paso siguiente

[Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)

Paso siguiente

[Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)

Paso 2: Configuración de la AWS Command Line Interface (AWS CLI)

En este paso, debe descargar y configurar la AWS CLI para usarla con Managed Service para Apache Flink.

Note

En los ejercicios introductorios de esta guía se presupone que está utilizando las credenciales de administrador (`adminuser`) en su cuenta para realizar las operaciones.

Note

Si ya tiene la AWS CLI instalada, es posible que tenga que actualizarla para obtener las últimas funcionalidades. Para obtener más información, consulte [Instalar la AWS Command Line Interface](#) en la Guía del usuario de AWS Command Line Interface. Ejecute el siguiente comando para comprobar la versión de la AWS CLI:

```
aws --version
```

Los ejercicios de este tutorial requieren la siguiente versión de la AWS CLI o posterior:

```
aws-cli/1.16.63
```

Para configurar la AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de la AWS Command Line Interface:
 - [Instalación de la AWS Command Line Interface](#)
 - [Configuración de la AWS CLI](#)
2. Añada un perfil con nombre para el usuario administrador en el archivo `config` de la AWS CLI. Puede utilizar este perfil cuando ejecute los comandos de la AWS CLI. Para obtener más información sobre los perfiles con nombre, consulte [Perfiles con nombre](#) en la Guía del usuario de AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Para ver una lista de las regiones de AWS disponibles, consulte [Regiones y puntos de conexión](#) en la Referencia general de Amazon Web Services.

Note

El código y los comandos de ejemplo de este tutorial utilizan la región Oeste de EE. UU. (Oregón). Para usar una región diferente, cambie la región en el código y los comandos de este tutorial por la región que desea usar.

3. Verifique la configuración introduciendo el siguiente comando de ayuda en el símbolo del sistema:

```
aws help
```

Tras configurar una AWS cuentaAWS CLI, puede realizar el siguiente ejercicio, en el que configurará una aplicación de ejemplo y probará la end-to-end configuración.

Paso siguiente

[Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)

Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink

En este ejercicio, deberá crear una aplicación de Managed Service para Apache Flink con flujos de datos como origen y receptor.

Esta sección contiene los siguientes pasos:

- [Crear dos Amazon Kinesis Data Streams](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Paso siguiente](#)

Crear dos Amazon Kinesis Data Streams

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, cree dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`). Su aplicación utiliza estos flujos para los flujos de origen y destino de la aplicación.

Puede crear estos flujos mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para crear flujos de datos (AWS CLI)

1. Para crear el primer flujo (`ExampleInputStream`), utilice el siguiente comando de la AWS CLI `create-stream` de Amazon Kinesis.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

2. Para crear el segundo flujo que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a `ExampleOutputStream`.

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3
STREAM_NAME = "ExampleInputStream"
def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Más adelante en el tutorial, se ejecuta el script `stock.py` para enviar datos a la aplicación.

```
$ python stock.py
```

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo [Project Object Model \(pom.xml\)](#) contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.java` contiene el método `main` que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos usando un objeto `StreamExecutionEnvironment`.
- La aplicación crea conectores de origen y recepción mediante propiedades estáticas. Para utilizar propiedades dinámicas de la aplicación, utilice los métodos `createSourceFromApplicationProperties` y `createSinkFromApplicationProperties` para crear los conectores. Estos métodos leen las propiedades de la aplicación para configurar los conectores.

Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar el código de la aplicación


En esta sección, se utiliza el compilador Apache Maven para crear el código de Java para la aplicación. Para obtener más información sobre la instalación de Apache Maven y el Java Development Kit (JDK), consulte [Requisitos previos para completar los ejercicios](#).

Para compilar el código de la aplicación

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar el código de una de las dos formas siguientes:
 - Utilice la herramienta de línea de comandos de Maven. Cree su archivo JAR ejecutando el siguiente comando en el directorio que contiene el archivo `pom.xml`:

```
mvn package -Dflink.version=1.13.2
```

- Use el entorno de desarrollo. Consulte la documentación de su entorno de desarrollo para obtener más información.

 Note

El código fuente proporcionado se basa en bibliotecas de Java 11.

Puede cargar el paquete como un archivo JAR o puede comprimir el paquete y cargarlo como un archivo ZIP. Si crea su aplicación mediante la AWS CLI, debe especificar el tipo de contenido del código (JAR o ZIP).

2. Si hay errores al compilar, verifique que la variable de entorno `JAVA_HOME` se ha configurado correctamente.

Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Cargar el código de Java de streaming de Apache Flink

En esta sección, creará un bucket de Amazon Simple Storage Service (Amazon S3) y cargará el código de la aplicación.

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket.
3. Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. En la consola de Amazon S3, elija el `<username>`bucket `ka-app-code-` y elija Upload.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior. Elija Siguiente.

9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Puede crear y ejecutar una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI.

Note

Cuando crea la aplicación mediante la consola, sus recursos AWS Identity and Access Management (de IAM) y de Amazon CloudWatch Logs se crean automáticamente. Si crea la aplicación mediante la AWS CLI, debe crear estos recursos por separado.

Temas

- [Crear y ejecutar la aplicación \(consola\)](#)
- [Crear y ejecutar la aplicación \(AWS CLI\)](#)

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje el menú desplegable de versión como Apache Flink versión 1.13.

4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **`kinesis-analytics-service-MyApplication-us-west-2`** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
```

```

        "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
    ],
    {
        "Sid": "DescribeLogGroups",
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogGroups"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:*"
        ]
    },
    {
        "Sid": "DescribeLogStreams",
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogStreams"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        ]
    },
    {
        "Sid": "PutLogEvents",
        "Effect": "Allow",
        "Action": [
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        ]
    },
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {

```

```

        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
]
}

```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Introduzca lo siguiente:

ID de grupo	Clave	Valor
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

5. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
6. Para el CloudWatch registro, active la casilla Activar.
7. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

En la MyApplication página, selecciona Detener. Confirme la acción.

Actualizar la aplicación

Mediante la consola, puede actualizar la configuración de la aplicación, tal como sus propiedades, ajustes de monitorización y la ubicación o el nombre de archivo JAR de la aplicación. También puede volver a cargar el JAR de la aplicación del bucket de Amazon S3 si necesita actualizar el código de la aplicación.

En la MyApplication página, elija Configurar. Actualice la configuración de la aplicación y elija Actualizar.

Crear y ejecutar la aplicación (AWS CLI)

En esta sección, se utiliza AWS CLI para crear y ejecutar la aplicación Managed Service para Apache Flink. Managed Service para Apache Flink utiliza el comando de la AWS CLI de `kinesisanalyticsv2` para crear e interactuar con las aplicaciones de Managed Service para Apache Flink.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción `read` en el flujo de origen y otra que concede permisos para las acciones `write` en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

`AKReadStreamWriteSinkStream`. Reemplace *username* por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Sustituya el ID de la cuenta en los nombres de recurso de Amazon (ARN) (*012345678901*) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username",
        "arn:aws:s3:::ka-app-code-username/*"
      ]
    },
    {
      "Sid": "ReadStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
    }
  ]
}
```

```
{
  "Sid": "WriteOutputStream",
  "Effect": "Allow",
  "Action": "kinesis:*",
  "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
}
```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial sobre cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Note

Para acceder a otros servicios de Amazon, puede usar AWS SDK for Java. Managed Service para Apache Flink establece automáticamente las credenciales requeridas por el SDK con las del rol de IAM de ejecución del servicio asociada a su aplicación. No hace falta realizar ningún otro paso.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Roles, Crear rol.


3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS. En Elegir el servicio que usará este rol, elija Kinesis. En Seleccionar su caso de uso, elija Kinesis Analytics.

Elija Siguiente: permisos.

4. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
5. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos del rol.

6. Asocie la política de permisos al rol.

 Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Asocie la política que ha creado en el paso anterior, [the section called “Crear una política de permisos”](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la ReadSourceStreamWriteSinkStream política AK y elija Adjuntar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación de Managed Service para Apache Flink

1. Copie el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del

bucket (*username*) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de ejemplo (*012345678901*) del rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```

2. Ejecute la acción [CreateApplication](#) con la solicitud anterior para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{
  "ApplicationName": "test",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

2. Ejecute la acción [StartApplication](#) con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "test"
```

```
}
```

2. Ejecute la acción [StopApplication](#) con la siguiente solicitud para detener la aplicación:

```
aws kinesisanalyticstv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso CloudWatch de Logs con su aplicación, consulte [the section called "Configuración del registro"](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
```

```
        "aws.region" : "us-west-2"
      }
    }
  ]
}
}
```

2. Ejecute la acción [UpdateApplication](#) con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción [UpdateApplication](#) de la AWS CLI.

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [the section called “Crear dos Amazon Kinesis Data Streams”](#).

```
{
```



```
"ApplicationName": "test",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
  "ApplicationCodeConfigurationUpdate": {
    "CodeContentUpdate": {
      "S3ContentLocationUpdate": {
        "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
        "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
        "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvpDU"
      }
    }
  }
}
```

Paso siguiente

[Paso 4: Eliminar recursos de AWS](#)

Paso 4: Eliminar recursos de AWS

En esta sección se incluyen los procedimientos para eliminar los recursos de AWS creados en el tutorial de Introducción.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)
- [Paso siguiente](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Paso siguiente


[Paso 5: Sigüientes pasos](#)

Paso 5: Sigüientes pasos

Ahora que ha creado y ejecutado una aplicación básica de Managed Service para Apache Flink, consulte los sigüientes recursos para conocer soluciones más avanzadas de Managed Service para Apache Flink.

- [Solución de datos de streaming de AWS para Amazon Kinesis](#): La solución de datos de streaming de AWS para Amazon Kinesis configura automáticamente los servicios de AWS necesarios para capturar, almacenar, procesar y entregar fácilmente datos de streaming. La solución ofrece varias opciones para resolver casos de uso de datos de streaming. La opción Managed Service for Apache Flink proporciona un ejemplo de ETL en end-to-end streaming que muestra una aplicación real que ejecuta operaciones analíticas con datos simulados de taxis de Nueva York. La solución configura todos los AWS recursos necesarios, como las funciones y políticas de IAM, un CloudWatch panel de control y alarmas. CloudWatch
- [Solución de datos de streaming de AWS para Amazon MSK](#): La solución de datos de streaming de AWS para Amazon MSK proporciona plantillas de AWS CloudFormation en las que los datos fluyen a través de los productores, el almacenamiento de streaming, los consumidores y los destinos.
- [Clickstream Lab con Apache Flink y Apache Kafka](#): Un laboratorio integral para casos de uso de secuencias de clics que utiliza Amazon Managed Streaming para Apache Kafka para almacenamiento de streaming y aplicaciones de Managed Service para Apache Flink para procesamiento de flujos.
- [Taller de Amazon Managed Service para Apache Flink](#): en este taller, crearás una arquitectura de end-to-end streaming para ingerir, analizar y visualizar datos de streaming prácticamente en tiempo real. Usted se propuso mejorar las operaciones de una empresa de taxis de la ciudad de Nueva York. Usted analiza los datos de telemetría de una flota de taxis de la ciudad de Nueva York prácticamente en tiempo real para optimizar las operaciones de su flota.
- [Managed Service para Apache Flink: ejemplos](#): En esta sección de esta Guía para desarrolladores se proporcionan ejemplos de cómo crear y trabajar con aplicaciones en Managed Service para Apache Flink. Incluyen ejemplos de código e step-by-step instrucciones que le ayudarán a crear un servicio gestionado para las aplicaciones de Apache Flink y a comprobar sus resultados.

- [Aprenda Flink: formación práctica](#): formación oficial introductoria sobre Apache Flink que le permitirá empezar a crear aplicaciones escalables de ETL de streaming, análisis y basadas en eventos.

 Note

Tenga en cuenta que Managed Service para Apache Flink no es compatible con la versión de Apache Flink (1.12) utilizada en esta formación. Puede usar Flink 1.15.2 en Flink Managed Service para Apache Flink.

Introducción: Flink 1.11.1

Este tema contiene una versión del [Cómo empezar \(DataStream API\)](#) tutorial que usa Apache Flink 1.11.1.

En esta sección, se presentan los conceptos fundamentales del servicio gestionado para Apache Flink y la DataStream API. Describe las opciones disponibles para crear y probar sus aplicaciones. También proporciona instrucciones para instalar las herramientas necesarias para completar los tutoriales de esta guía y crear su primera aplicación.

Temas

- [Componentes de una aplicación de Managed Service para Apache Flink](#)
- [Requisitos previos para completar los ejercicios](#)
- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)
- [Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)
- [Paso 4: Eliminar recursos de AWS](#)
- [Paso 5: Sigüientes pasos](#)

Componentes de una aplicación de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Java/Apache Maven o Scala que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

Una aplicación de Managed Service para Apache Flink tiene los siguientes componentes:

- **Propiedades de tiempo de ejecución:** puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- **Fuente:** la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un bucket de Amazon S3, etc. Para obtener más información, consulte [Orígenes](#).
- **Operadores:** la aplicación procesa los datos mediante uno o más operadores. Un operador puede transformar, enriquecer o agregar datos. Para obtener más información, consulte [Operadores de la API de DataStream](#).
- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores](#).

Después de crear, compilar y empaquetar el código de la aplicación, debe cargar el paquete del código a un bucket de Amazon Simple Storage Service (Amazon S3). Luego debe crear la aplicación de Managed Service para Apache Flink. Introduzca la ubicación del paquete del código, un flujo de datos de Kinesis como origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos para completar los ejercicios

Para completar los pasos de esta guía, debe disponer de lo siguiente:

- [Java Development Kit \(JDK\), versión 11](#). Establezca la variable de entorno JAVA_HOME para señalar la ubicación de la instalación del JDK.
- Le recomendamos utilizar un entorno de desarrollo (como [Eclipse Java Neon](#) o [IntelliJ Idea](#)) para desarrollar y compilar su aplicación.
- [Cliente Git](#). Si aún no lo ha hecho, instale el cliente Git.
- [Apache Maven Compiler Plugin](#). Maven debe estar en su ruta de trabajo. Para probar la instalación de Apache Maven, introduzca lo siguiente:

```
$ mvn -version
```

Para empezar, vaya a [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#).

Paso 1: Configurar una cuenta de AWS y crear un usuario administrador

Registro para obtener una Cuenta de AWS

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

Cómo registrarse para una Cuenta de AWS

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

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

Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, [asigne acceso administrativo a un usuario administrativo](#) y utilice únicamente el usuario raíz para realizar [tareas que requieran acceso de usuario raíz](#).

AWS le enviará un correo electrónico de confirmación luego de completar el proceso de registro.

Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en <https://aws.amazon.com/> y seleccionando Mi cuenta.

Crear un usuario administrativo

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

Protección de su Usuario raíz de la cuenta de AWS

1. Inicie sesión en la [AWS Management Console](#) como propietario de cuenta, elija Usuario raíz e ingrese el email de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Signing in as the root user](#) en la Guía del usuario de AWS Sign-In.

2. Active la autenticación multifactor (MFA) para el usuario raíz.

Para obtener instrucciones, consulte [Habilitar un dispositivo MFA virtual para el usuario raíz de la Cuenta de AWS \(consola\)](#) en la Guía del usuario de IAM.

Creación de un usuario administrativo

1. Activar IAM Identity Center

Para conocer las instrucciones, consulte [Habilitar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center.

2. En IAM Identity Center, otorga acceso administrativo a un usuario administrativo.

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

Cómo iniciar sesión como usuario administrativo

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

Para obtener ayuda para iniciar sesión con un usuario del IAM Identity Center, consulte [Iniciar sesión en el portal de acceso de AWS](#) en la Guía del Usuario de AWS Sign-In.

Conceder acceso programático

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

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

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • En AWS CLI, consulte Configuración de la AWS CLI para usar AWS IAM Identity Center en la

¿Qué usuario necesita acceso programático?	Para	Mediante
		<p>Guía del usuario de AWS Command Line Interface.</p> <ul style="list-style-type: none">• En los SDK de AWS, las herramientas y las API de AWS, consulte Autenticación de IAM Identity Center en la Guía de referencia de SDK de AWS y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del Usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para los SDK de AWS y las herramientas, consulte Autenticación mediante credenciales a largo plazo en la Guía de referencia de SDK de AWS y herramientas. • Para las API de AWS, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Paso siguiente

[Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)

Paso 2: Configuración de la AWS Command Line Interface (AWS CLI)

En este paso, debe descargar y configurar la AWS CLI para usarla con Managed Service para Apache Flink.

Note

En los ejercicios introductorios de esta guía se presupone que está utilizando las credenciales de administrador (`adminuser`) en su cuenta para realizar las operaciones.

Note

Si ya tiene la AWS CLI instalada, es posible que tenga que actualizarla para obtener las últimas funcionalidades. Para obtener más información, consulte [Instalar la AWS Command Line Interface](#) en la Guía del usuario de AWS Command Line Interface. Ejecute el siguiente comando para comprobar la versión de la AWS CLI:

```
aws --version
```

Los ejercicios de este tutorial requieren la siguiente versión de la AWS CLI o posterior:

```
aws-cli/1.16.63
```

Para configurar la AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de la AWS Command Line Interface:
 - [Instalación de la AWS Command Line Interface](#)
 - [Configuración de la AWS CLI](#)
2. Añada un perfil con nombre para el usuario administrador en el archivo `config` de la AWS CLI. Puede utilizar este perfil cuando ejecute los comandos de la AWS CLI. Para obtener más información sobre los perfiles con nombre, consulte [Perfiles con nombre](#) en la Guía del usuario de AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Para ver una lista de las regiones de AWS disponibles, consulte [Regiones y puntos de conexión](#) en la Referencia general de Amazon Web Services.

Note

El código y los comandos de ejemplo de este tutorial utilizan la región Oeste de EE. UU. (Oregón). Para usar una región diferente, cambie la región en el código y los comandos de este tutorial por la región que desea usar.

3. Verifique la configuración introduciendo el siguiente comando de ayuda en el símbolo del sistema:

```
aws help
```

Tras configurar una AWS cuentaAWS CLI, puede realizar el siguiente ejercicio, en el que configurará una aplicación de ejemplo y probará la end-to-end configuración.

Paso siguiente

[Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)

Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink

En este ejercicio, deberá crear una aplicación de Managed Service para Apache Flink con flujos de datos como origen y receptor.

Esta sección contiene los siguientes pasos:

- [Crear dos Amazon Kinesis Data Streams](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Paso siguiente](#)

Crear dos Amazon Kinesis Data Streams

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, cree dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`). Su aplicación utiliza estos flujos para los flujos de origen y destino de la aplicación.

Puede crear estos flujos mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para crear flujos de datos (AWS CLI)

1. Para crear el primer flujo (`ExampleInputStream`), utilice el siguiente comando de la AWS CLI `create-stream` de Amazon Kinesis.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

2. Para crear el segundo flujo que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a `ExampleOutputStream`.

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
            PartitionKey="partitionkey"
        )

if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Más adelante en el tutorial, se ejecuta el script `stock.py` para enviar datos a la aplicación.

```
$ python stock.py
```

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo [Project Object Model \(pom.xml\)](#) contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.java` contiene el método `main` que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos usando un objeto `StreamExecutionEnvironment`.
- La aplicación crea conectores de origen y recepción mediante propiedades estáticas. Para utilizar propiedades dinámicas de la aplicación, utilice los métodos `createSourceFromApplicationProperties` y `createSinkFromApplicationProperties` para crear los conectores. Estos métodos leen las propiedades de la aplicación para configurar los conectores.

Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar el código de la aplicación

En esta sección, se utiliza el compilador Apache Maven para crear el código de Java para la aplicación. Para obtener más información sobre la instalación de Apache Maven y el Java Development Kit (JDK), consulte [Requisitos previos para completar los ejercicios](#).

Para compilar el código de la aplicación

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar el código de una de las dos formas siguientes:

- Utilice la herramienta de línea de comandos de Maven. Cree su archivo JAR ejecutando el siguiente comando en el directorio que contiene el archivo `pom.xml`:

```
mvn package -Dflink.version=1.11.3
```

- Use el entorno de desarrollo. Consulte la documentación de su entorno de desarrollo para obtener más información.

Note

El código fuente proporcionado se basa en bibliotecas de Java 11. Asegúrese de que la versión Java de su proyecto sea 11.

Puede cargar el paquete como un archivo JAR o puede comprimir el paquete y cargarlo como un archivo ZIP. Si crea su aplicación mediante la AWS CLI, debe especificar el tipo de contenido del código (JAR o ZIP).

2. Si hay errores al compilar, verifique que la variable de entorno `JAVA_HOME` se ha configurado correctamente.

Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Cargar el código de Java de streaming de Apache Flink

En esta sección, creará un bucket de Amazon Simple Storage Service (Amazon S3) y cargará el código de la aplicación.

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket.

3. Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. En la consola de Amazon S3, elija el *<username>*bucket ka-app-code- y elija Upload.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior. Elija Siguiente.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Puede crear y ejecutar una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI.

Note

Cuando crea la aplicación mediante la consola, sus recursos AWS Identity and Access Management (de IAM) y de Amazon CloudWatch Logs se crean automáticamente. Si crea la aplicación mediante la AWS CLI, debe crear estos recursos por separado.

Temas

- [Crear y ejecutar la aplicación \(consola\)](#)
- [Crear y ejecutar la aplicación \(AWS CLI\)](#)

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>

2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje el menú desplegable de versión como Apache Flink versión 1.11 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "ReadCode",
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "Resource": [
      "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
    ]
  },
  {
    "Sid": "DescribeLogGroups",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
  },
  {
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  }
]
```

```

    ],
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    }
  ]
}

```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. En Propiedades, en ID de grupo, escriba **ProducerConfigProperties**.
5. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
ProducerConfigProperties	flink.inputstream.initpos	LATEST

ID de grupo	Clave	Valor
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

6. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
7. Para el CloudWatch registro, active la casilla Activar.
8. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Ejecutar la aplicación

Para ver el gráfico de trabajos de Flink, ejecute la aplicación, abra el panel de Apache Flink y elija el trabajo de Flink que desee.

Detener la aplicación

En la MyApplication página, selecciona Detener. Confirme la acción.

Actualizar la aplicación

Mediante la consola, puede actualizar la configuración de la aplicación, tal como sus propiedades, ajustes de monitorización y la ubicación o el nombre de archivo JAR de la aplicación. También puede volver a cargar el JAR de la aplicación del bucket de Amazon S3 si necesita actualizar el código de la aplicación.

En la MyApplication página, elija Configurar. Actualice la configuración de la aplicación y elija Actualizar.

Crear y ejecutar la aplicación (AWS CLI)

En esta sección, se utiliza AWS CLI para crear y ejecutar la aplicación Managed Service para Apache Flink. Managed Service para Apache Flink utiliza el `kinesisanalyticsv2` AWS CLI comando para crear aplicaciones Managed Service para Apache Flink e interactuar con ellas.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción `read` en el flujo de origen y otra que concede permisos para las acciones `write` en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

`AKReadSourceStreamWriteSinkStream`. Reemplace *username* por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Sustituya el ID de la cuenta en los nombres de recurso de Amazon (ARN) (*012345678901*) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username",
        "arn:aws:s3:::ka-app-code-username/*"
      ]
    }
  ]
}
```

```
    ],
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial sobre cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Note

Para acceder a otros servicios de Amazon, puede usar AWS SDK for Java. Managed Service para Apache Flink establece automáticamente las credenciales requeridas por el SDK con las del rol de IAM de ejecución del servicio asociada a su aplicación. No hace falta realizar ningún otro paso.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM


1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Roles, Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS. En Elegir el servicio que usará este rol, elija Kinesis. En Seleccionar su caso de uso, elija Kinesis Analytics.

Elija Siguiente: permisos.

4. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
5. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos del rol.

6. Asocie la política de permisos al rol.

 Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Asocie la política que ha creado en el paso anterior, [the section called “Crear una política de permisos”](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la ReadSourceStreamWriteSinkStream política AK y elija Adjuntar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación de Managed Service para Apache Flink

1. Copie el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (*username*) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de ejemplo (*012345678901*) del rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_11",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```


2. Ejecute la acción [CreateApplication](#) con la solicitud anterior para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://  
create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{  
  "ApplicationName": "test",  
  "RunConfiguration": {  
    "ApplicationRestoreConfiguration": {  
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"  
    }  
  }  
}
```

2. Ejecute la acción [StartApplication](#) con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Ejecute la acción [StopApplication](#) con la siguiente solicitud para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso CloudWatch de Logs con su aplicación, consulte [the section called “Configuración del registro”](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        }
      ]
    }
  }
}
```

```
    },
    {
      "PropertyGroupId": "ConsumerConfigProperties",
      "PropertyMap" : {
        "aws.region" : "us-west-2"
      }
    }
  ]
}
}
```

2. Ejecute la acción [UpdateApplication](#) con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción [UpdateApplication](#) de la AWS CLI.

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket

(<username>) con el sufijo que haya elegido en la sección [the section called “Crear dos Amazon Kinesis Data Streams”](#).

```
{
  "ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

Paso siguiente

[Paso 4: Eliminar recursos de AWS](#)

Paso 4: Eliminar recursos de AWS

En esta sección se incluyen los procedimientos para eliminar los recursos de AWS creados en el tutorial de Introducción.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)
- [Paso siguiente](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.

2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.

3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Paso siguiente

[Paso 5: Sigüientes pasos](#)

Paso 5: Sigüientes pasos

Ahora que ha creado y ejecutado una aplicación básica de Managed Service para Apache Flink, consulte los sigüientes recursos para conocer soluciones más avanzadas de Managed Service para Apache Flink.

- [Solución de datos de streaming de AWS para Amazon Kinesis](#): La solución de datos de streaming de AWS para Amazon Kinesis configura automáticamente los servicios de AWS necesarios para capturar, almacenar, procesar y entregar fácilmente datos de streaming. La solución ofrece varias opciones para resolver casos de uso de datos de streaming. La opción Managed Service for Apache Flink proporciona un ejemplo de ETL en end-to-end streaming que muestra una aplicación real que ejecuta operaciones analíticas con datos simulados de taxis de Nueva York. La solución configura todos los AWS recursos necesarios, como las funciones y políticas de IAM, un CloudWatch panel de control y alarmas. CloudWatch
- [Solución de datos de streaming de AWS para Amazon MSK](#): La solución de datos de streaming de AWS para Amazon MSK proporciona plantillas de AWS CloudFormation en las que los datos fluyen a través de los productores, el almacenamiento de streaming, los consumidores y los destinos.
- [Clickstream Lab con Apache Flink y Apache Kafka](#): Un laboratorio integral para casos de uso de secuencias de clics que utiliza Amazon Managed Streaming para Apache Kafka para almacenamiento de streaming y aplicaciones de Managed Service para Apache Flink para procesamiento de flujos.
- [Taller de Amazon Managed Service para Apache Flink](#): en este taller, crearás una arquitectura de end-to-end streaming para ingerir, analizar y visualizar datos de streaming prácticamente en tiempo real. Usted se propuso mejorar las operaciones de una empresa de taxis de la ciudad de Nueva York. Usted analiza los datos de telemetría de una flota de taxis de la ciudad de Nueva York prácticamente en tiempo real para optimizar las operaciones de su flota.
- [Managed Service para Apache Flink: ejemplos](#): En esta sección de esta Guía para desarrolladores se proporcionan ejemplos de cómo crear y trabajar con aplicaciones en Managed Service para

Apache Flink. Incluyen ejemplos de código e step-by-step instrucciones que le ayudarán a crear un servicio gestionado para las aplicaciones de Apache Flink y a comprobar sus resultados.

- [Aprenda Flink: formación práctica](#): formación oficial introductoria sobre Apache Flink que le permitirá empezar a crear aplicaciones escalables de ETL de streaming, análisis y basadas en eventos.

Note

Tenga en cuenta que Managed Service para Apache Flink no es compatible con la versión de Apache Flink (1.12) utilizada en esta formación. Puede usar Flink 1.15.2 en Flink Managed Service para Apache Flink.

- Ejemplos de [código de Apache Flink: GitHub repositorio de una amplia variedad de ejemplos](#) de aplicaciones de Apache Flink.

Introducción: Flink 1.8.2

Este tema contiene una versión del Tutorial [Cómo empezar \(DataStream API\)](#) que usa Apache Flink 1.8.2.

Temas

- [Componentes de la aplicación de Managed Service para Apache Flink](#)
- [Requisitos previos para completar los ejercicios](#)
- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)
- [Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)
- [Paso 4: Eliminar recursos de AWS](#)

Componentes de la aplicación de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Java/Apache Maven o Scala que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

Una aplicación de Managed Service para Apache Flink tiene los siguientes componentes:

- **Propiedades de tiempo de ejecución:** puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- **Fuente:** la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un bucket de Amazon S3, etc. Para obtener más información, consulte [Orígenes](#).
- **Operadores:** la aplicación procesa los datos mediante uno o más operadores. Un operador puede transformar, enriquecer o agregar datos. Para obtener más información, consulte [Operadores de la API de DataStream](#).
- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores](#).

Después de crear, compilar y empaquetar el código de la aplicación, debe cargar el paquete del código a un bucket de Amazon Simple Storage Service (Amazon S3). Luego debe crear la aplicación de Managed Service para Apache Flink. Introduzca la ubicación del paquete del código, un flujo de datos de Kinesis como origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos para completar los ejercicios

Para completar los pasos de esta guía, debe disponer de lo siguiente:

- [Java Development Kit](#) (JDK), versión 8. Establezca la variable de entorno JAVA_HOME para señalar la ubicación de la instalación del JDK.
- Para usar el conector Apache Flink Kinesis en este tutorial, debe descargar e instalar Apache Flink. Para obtener más detalles, consulte [Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink](#).
- Le recomendamos utilizar un entorno de desarrollo (como [Eclipse Java Neon](#) o [IntelliJ Idea](#)) para desarrollar y compilar su aplicación.
- [Cliente Git](#). Si aún no lo ha hecho, instale el cliente Git.
- [Apache Maven Compiler Plugin](#). Maven debe estar en su ruta de trabajo. Para probar la instalación de Apache Maven, introduzca lo siguiente:

```
$ mvn -version
```


Para empezar, vaya a [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#).

Paso 1: Configurar una cuenta de AWS y crear un usuario administrador

Registro para obtener una Cuenta de AWS

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

Cómo registrarse para una Cuenta de AWS

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

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

Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, [asigne acceso administrativo a un usuario administrativo](#) y utilice únicamente el usuario raíz para realizar [tareas que requieran acceso de usuario raíz](#).

AWS le enviará un correo electrónico de confirmación luego de completar el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en <https://aws.amazon.com/> y seleccionando Mi cuenta.

Crear un usuario administrativo

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

Protección de su Usuario raíz de la cuenta de AWS

1. Inicie sesión en la [AWS Management Console](#) como propietario de cuenta, elija Usuario raíz e ingrese el email de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Signing in as the root user](#) en la Guía del usuario de AWS Sign-In.

2. Active la autenticación multifactor (MFA) para el usuario raíz.

Para obtener instrucciones, consulte [Habilitar un dispositivo MFA virtual para el usuario raíz de la Cuenta de AWS \(consola\)](#) en la Guía del usuario de IAM.

Creación de un usuario administrativo

1. Activar IAM Identity Center

Para conocer las instrucciones, consulte [Habilitar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center.

2. En IAM Identity Center, otorga acceso administrativo a un usuario administrativo.

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

Cómo iniciar sesión como usuario administrativo

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

Para obtener ayuda para iniciar sesión con un usuario del IAM Identity Center, consulte [Iniciar sesión en el portal de acceso de AWS](#) en la Guía del Usuario de AWS Sign-In.

Conceder acceso programático

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

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

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI,	Siga las instrucciones de la interfaz que desea utilizar:

¿Qué usuario necesita acceso programático?	Para	Mediante
(Usuarios administrados en IAM Identity Center)	los SDK de AWS o las API de AWS.	<ul style="list-style-type: none">• En AWS CLI, consulte Configuración de la AWS CLI para usar AWS IAM Identity Center en la Guía del usuario de AWS Command Line Interface.• En los SDK de AWS, las herramientas y las API de AWS, consulte Autenticación de IAM Identity Center en la Guía de referencia de SDK de AWS y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del Usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para los SDK de AWS y las herramientas, consulte Autenticación mediante credenciales a largo plazo en la Guía de referencia de SDK de AWS y herramientas. • Para las API de AWS, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Paso 2: Configuración de la AWS Command Line Interface (AWS CLI)

En este paso, debe descargar y configurar la AWS CLI para usarla con Managed Service para Apache Flink.

Note

En los ejercicios introductorios de esta guía se presupone que está utilizando las credenciales de administrador (`adminuser`) en su cuenta para realizar las operaciones.

Note

Si ya tiene la AWS CLI instalada, es posible que tenga que actualizarla para obtener las últimas funcionalidades. Para obtener más información, consulte [Instalar la AWS Command Line Interface](#) en la Guía del usuario de AWS Command Line Interface. Ejecute el siguiente comando para comprobar la versión de la AWS CLI:

```
aws --version
```

Los ejercicios de este tutorial requieren la siguiente versión de la AWS CLI o posterior:

```
aws-cli/1.16.63
```

Para configurar la AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de la AWS Command Line Interface:
 - [Instalación de la AWS Command Line Interface](#)
 - [Configuración de la AWS CLI](#)
2. Añada un perfil con nombre para el usuario administrador en el archivo config de la AWS CLI. Puede utilizar este perfil cuando ejecute los comandos de la AWS CLI. Para obtener más información sobre los perfiles con nombre, consulte [Perfiles con nombre](#) en la Guía del usuario de AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Para ver una lista de las regiones disponibles, consulte este artículo sobre las [regiones y puntos de enlace de](#) en la Referencia general de Amazon Web Services.

Note

El código y los comandos de ejemplo de este tutorial utilizan la región Oeste de EE. UU. (Oregón). Para usar una región de AWS diferente, cambie la región en el código y los comandos de este tutorial por la región que desea usar.

3. Verifique la configuración introduciendo el siguiente comando de ayuda en el símbolo del sistema:

```
aws help
```

Tras configurar una AWS cuenta y el AWS CLI, puede probar el siguiente ejercicio, en el que configurará una aplicación de ejemplo y probará la end-to-end configuración.

Paso siguiente

[Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)

Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink

En este ejercicio, deberá crear una aplicación de Managed Service para Apache Flink con flujos de datos como origen y receptor.

Esta sección contiene los siguientes pasos:

- [Crear dos Amazon Kinesis Data Streams](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)
- [Paso siguiente](#)

Crear dos Amazon Kinesis Data Streams

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, cree dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`). Su aplicación utiliza estos flujos para los flujos de origen y destino de la aplicación.

Puede crear estos flujos mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para crear flujos de datos (AWS CLI)

1. Para crear el primer flujo (`ExampleInputStream`), utilice el siguiente comando de la AWS CLI `create-stream` de Amazon Kinesis.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

2. Para crear el segundo flujo que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a `ExampleOutputStream`.

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
            PartitionKey="partitionkey"
        )

if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Más adelante en el tutorial, se ejecuta el script `stock.py` para enviar datos a la aplicación.

```
$ python stock.py
```

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Clone el repositorio remoto con el siguiente comando:


```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo [Project Object Model \(pom.xml\)](#) contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.java` contiene el método `main` que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos usando un objeto `StreamExecutionEnvironment`.
- La aplicación crea conectores de origen y recepción mediante propiedades estáticas. Para utilizar propiedades dinámicas de la aplicación, utilice los métodos `createSourceFromApplicationProperties` y `createSinkFromApplicationProperties` para crear los conectores. Estos métodos leen las propiedades de la aplicación para configurar los conectores.

Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar el código de la aplicación

En esta sección, se utiliza el compilador Apache Maven para crear el código de Java para la aplicación. Para obtener más información sobre la instalación de Apache Maven y el Java Development Kit (JDK), consulte [Requisitos previos para completar los ejercicios](#).

Note

Para utilizar el conector de Kinesis con versiones de Apache Flink anteriores a la 1.11, debe descargar, compilar e instalar Apache Maven. Para obtener más información, consulte [the section called “Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink”](#).

Para compilar el código de la aplicación

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar el código de una de las dos formas siguientes:
 - Utilice la herramienta de línea de comandos de Maven. Cree su archivo JAR ejecutando el siguiente comando en el directorio que contiene el archivo `pom.xml`:

```
mvn package -Dflink.version=1.8.2
```

- Use el entorno de desarrollo. Consulte la documentación de su entorno de desarrollo para obtener más información.

Note

El código fuente proporcionado se basa en bibliotecas de Java 1.8. Asegúrese de que la versión Java de su proyecto sea 1.8.

Puede cargar el paquete como un archivo JAR o puede comprimir el paquete y cargarlo como un archivo ZIP. Si crea su aplicación mediante la AWS CLI, debe especificar el tipo de contenido del código (JAR o ZIP).

2. Si hay errores al compilar, verifique que la variable de entorno `JAVA_HOME` se ha configurado correctamente.

Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Cargar el código de Java de streaming de Apache Flink

En esta sección, creará un bucket de Amazon Simple Storage Service (Amazon S3) y cargará el código de la aplicación.

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket.
3. Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. En la consola de Amazon S3, elija el *<username>*bucket ka-app-code- y, a continuación, seleccione Upload.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior. Elija Siguiente.
9. No es necesario cambiar ninguno de los ajustes del objeto, por lo tanto, elija Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Puede crear y ejecutar una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI.

Note

Cuando crea la aplicación mediante la consola, sus recursos AWS Identity and Access Management (de IAM) y de Amazon CloudWatch Logs se crean automáticamente. Si crea la aplicación mediante la AWS CLI, debe crear estos recursos por separado.

Temas

- [Crear y ejecutar la aplicación \(consola\)](#)
- [Crear y ejecutar la aplicación \(AWS CLI\)](#)

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.
 - Deje el menú desplegable de versión como Apache Flink versión 1.8 (versión recomendada).
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política **kinesis-analytics-service-MyApplication-us-west-2** que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (**012345678901**) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
    }
  ]
}
```

```

    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.
 - En Ruta al objeto de Amazon S3, introduzca **aws-kinesis-analytics-java-apps-1.0.jar**.

3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

5. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
6. Para el CloudWatch registro, active la casilla Activar.
7. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Ejecutar la aplicación

1. En la MyApplication página, selecciona Ejecutar. Confirme la acción.
2. Cuando la aplicación se está ejecutando, actualice la página. La consola muestra el Gráfico de la aplicación.

Detener la aplicación

En la MyApplication página, selecciona Detener. Confirme la acción.

Actualizar la aplicación

Mediante la consola, puede actualizar la configuración de la aplicación, tal como sus propiedades, ajustes de monitorización y la ubicación o el nombre de archivo JAR de la aplicación. También puede volver a cargar el JAR de la aplicación del bucket de Amazon S3 si necesita actualizar el código de la aplicación.

En la MyApplication página, elija Configurar. Actualice la configuración de la aplicación y elija Actualizar.

Crear y ejecutar la aplicación (AWS CLI)

En esta sección, se utiliza AWS CLI para crear y ejecutar la aplicación Managed Service para Apache Flink. Managed Service para Apache Flink utiliza el comando de la AWS CLI de `kinesisanalyticsv2` para crear e interactuar con las aplicaciones de Managed Service para Apache Flink.

Crear una política de permisos

Note

Debe crear una política de permisos y un rol para su aplicación. Si no crea estos recursos de IAM, la aplicación no podrá acceder a sus flujos de datos y de registro.

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción `read` en el flujo de origen y otra que concede permisos para las acciones `write` en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

```
AKReadSourceStreamWriteSinkStream. Reemplace username por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Sustituya el ID de la cuenta en los nombres de recurso de Amazon (ARN) (012345678901) por el ID de su cuenta.
```



```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username",
        "arn:aws:s3:::ka-app-code-username/*"
      ]
    },
    {
      "Sid": "ReadInputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleOutputStream"
    }
  ]
}

```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial sobre cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Note

Para acceder a otros servicios de Amazon, puede usar AWS SDK for Java. Managed Service para Apache Flink establece automáticamente las credenciales requeridas por el SDK con las del rol de IAM de ejecución del servicio asociada a su aplicación. No hace falta realizar ningún otro paso.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Roles, Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS. En Elegir el servicio que usará este rol, elija Kinesis. En Seleccionar su caso de uso, elija Kinesis Analytics.

Elija Siguiente: permisos.

4. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
5. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos del rol.

6. Asocie la política de permisos al rol.

Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Asocie la política que ha creado en el paso anterior, [the section called “Crear una política de permisos”](#).

- a. En la página Resumen, elija la pestaña Permisos.

- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la ReadSourceStreamWriteSinkStream política AK y elija Adjuntar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación de Managed Service para Apache Flink

1. Copie el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (*username*) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de ejemplo (*012345678901*) del rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_8",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "flink.stream.initpos": "LATEST",
            "aws.region": "us-west-2",
            "AggregationEnabled": "false"
          }
        }
      ]
    }
  }
}
```

```
    }
  },
  {
    "PropertyGroupId": "ConsumerConfigProperties",
    "PropertyMap" : {
      "aws.region" : "us-west-2"
    }
  }
]
}
}
```

2. Ejecute la acción [CreateApplication](#) con la solicitud anterior para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{
  "ApplicationName": "test",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

2. Ejecute la acción [StartApplication](#) con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Ejecute la acción [StopApplication](#) con la siguiente solicitud para detener la aplicación:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso CloudWatch de Logs con su aplicación, consulte [the section called “Configuración del registro”](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "test",
```

```
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
  "EnvironmentPropertyUpdates": {
    "PropertyGroups": [
      {
        "PropertyGroupId": "ProducerConfigProperties",
        "PropertyMap" : {
          "flink.stream.initpos" : "LATEST",
          "aws.region" : "us-west-2",
          "AggregationEnabled" : "false"
        }
      },
      {
        "PropertyGroupId": "ConsumerConfigProperties",
        "PropertyMap" : {
          "aws.region" : "us-west-2"
        }
      }
    ]
  }
}
```

2. Ejecute la acción [UpdateApplication](#) con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción [UpdateApplication](#) de la AWS CLI.

Note

Para cargar una nueva versión del código de la aplicación con el mismo nombre de archivo, debe especificar la nueva versión del objeto. Para obtener más información sobre el uso de versiones de objetos de Amazon S3, consulte [Habilitar o deshabilitar el control de versiones](#).

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [the section called “Crear dos Amazon Kinesis Data Streams”](#).

```
{
  "ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

Paso siguiente

[Paso 4: Eliminar recursos de AWS](#)

Paso 4: Eliminar recursos de AWS

En esta sección se incluyen los procedimientos para eliminar los recursos de AWS creados en el tutorial de Introducción.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)

- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. Elija Configurar.
4. En la sección Instantáneas, seleccione Deshabilitar y, a continuación, seleccione Actualizar.
5. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Kinesis Data Streams, ExampleInputStream elija.
3. En la ExampleInputStreampágina, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.

4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Introducción: Flink 1.6.2

Este tema contiene una versión del Tutorial [Cómo empezar \(DataStream API\)](#) que usa Apache Flink 1.6.2.

Temas

- [Componentes de Managed Service para Apache Flink](#)
- [Requisitos previos para completar los ejercicios](#)
- [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#)
- [Paso 2: Configuración de la AWS Command Line Interface \(AWS CLI\)](#)
- [Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)
- [Paso 4: Eliminar recursos de AWS](#)

Componentes de Managed Service para Apache Flink

Para procesar los datos, su aplicación de Managed Service para Apache Flink utiliza una aplicación Java/Apache Maven o Scala que procesa las entradas y produce las salidas mediante el tiempo de ejecución de Apache Flink.

Managed Service para Apache Flink tiene los siguientes componentes:

- **Propiedades de tiempo de ejecución:** puede usar las propiedades de tiempo de ejecución para configurar su aplicación sin tener que volver a compilar el código de la aplicación.
- **Fuente:** la aplicación consume datos mediante una fuente. Un conector de origen lee los datos de un flujo de datos de Kinesis, un bucket de Amazon S3, etc. Para obtener más información, consulte [Orígenes](#).
- **Operadores:** la aplicación procesa los datos mediante uno o más operadores. Un operador puede transformar, enriquecer o agregar datos. Para obtener más información, consulte [Operadores de la API de DataStream](#).
- **Receptor:** la aplicación produce datos para fuentes externas mediante el uso de receptores. Un conector de recepción escribe datos en un flujo de datos de Kinesis, un flujo de Kinesis Data Firehose, un bucket de Amazon S3, etc. Para obtener más información, consulte [Receptores](#).

Después de crear, compilar y empaquetar la aplicación, debe cargar el paquete del código a un bucket de Amazon Simple Storage Service (Amazon S3). Luego debe crear la aplicación de Managed Service para Apache Flink. Introduzca la ubicación del paquete del código, un flujo de datos de Kinesis como origen de datos de streaming y, normalmente, una ubicación de streaming o archivo que recibe los datos procesados de la aplicación.

Requisitos previos para completar los ejercicios

Para completar los pasos de esta guía, debe disponer de lo siguiente:

- [Java Development Kit](#) (JDK), versión 8. Establezca la variable de entorno JAVA_HOME para señalar la ubicación de la instalación del JDK.
- Le recomendamos utilizar un entorno de desarrollo (como [Eclipse Java Neon](#) o [IntelliJ Idea](#)) para desarrollar y compilar su aplicación.
- [Cliente Git](#). Si aún no lo ha hecho, instale el cliente Git.
- [Apache Maven Compiler Plugin](#). Maven debe estar en su ruta de trabajo. Para probar la instalación de Apache Maven, introduzca lo siguiente:

```
$ mvn -version
```

Para empezar, vaya a [Paso 1: Configurar una cuenta de AWS y crear un usuario administrador](#).

Paso 1: Configurar una cuenta de AWS y crear un usuario administrador

Registro para obtener una Cuenta de AWS

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

Cómo registrarse para una Cuenta de AWS

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

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

Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, [asigne acceso administrativo a un usuario administrativo](#) y utilice únicamente el usuario raíz para realizar [tareas que requieran acceso de usuario raíz](#).

AWS le enviará un correo electrónico de confirmación luego de completar el proceso de registro.

Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en <https://aws.amazon.com/> y seleccionando Mi cuenta.

Crear un usuario administrativo

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

Protección de su Usuario raíz de la cuenta de AWS

1. Inicie sesión en la [AWS Management Console](#) como propietario de cuenta, elija Usuario raíz e ingrese el email de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Signing in as the root user](#) en la Guía del usuario de AWS Sign-In.

2. Active la autenticación multifactor (MFA) para el usuario raíz.

Para obtener instrucciones, consulte [Habilitar un dispositivo MFA virtual para el usuario raíz de la Cuenta de AWS \(consola\)](#) en la Guía del usuario de IAM.

Creación de un usuario administrativo

1. Activar IAM Identity Center

Para conocer las instrucciones, consulte [Habilitar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center.

2. En IAM Identity Center, otorga acceso administrativo a un usuario administrativo.

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

Cómo iniciar sesión como usuario administrativo

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

Para obtener ayuda para iniciar sesión con un usuario del IAM Identity Center, consulte [Iniciar sesión en el portal de acceso de AWS](#) en la Guía del Usuario de AWS Sign-In.

Conceder acceso programático

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

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

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en IAM Identity Center)	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • En AWS CLI, consulte Configuración de la AWS CLI para usar AWS IAM Identity Center en la

¿Qué usuario necesita acceso programático?	Para	Mediante
		<p>Guía del usuario de AWS Command Line Interface.</p> <ul style="list-style-type: none">• En los SDK de AWS, las herramientas y las API de AWS, consulte Autenticación de IAM Identity Center en la Guía de referencia de SDK de AWS y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del Usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los SDK de AWS o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: <ul style="list-style-type: none"> • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para los SDK de AWS y las herramientas, consulte Autenticación mediante credenciales a largo plazo en la Guía de referencia de SDK de AWS y herramientas. • Para las API de AWS, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Paso 2: Configuración de la AWS Command Line Interface (AWS CLI)

En este paso, debe descargar y configurar la AWS CLI para usarla con Managed Service para Apache Flink.

Note

En los ejercicios introductorios de esta guía se presupone que está utilizando las credenciales de administrador (`adminuser`) en su cuenta para realizar las operaciones.

Note

Si ya tiene la AWS CLI instalada, es posible que tenga que actualizarla para obtener las últimas funcionalidades. Para obtener más información, consulte [Instalar la AWS Command Line Interface](#) en la Guía del usuario de AWS Command Line Interface. Ejecute el siguiente comando para comprobar la versión de la AWS CLI:

```
aws --version
```

Los ejercicios de este tutorial requieren la siguiente versión de la AWS CLI o posterior:

```
aws-cli/1.16.63
```

Para configurar la AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de la AWS Command Line Interface:
 - [Instalación de la AWS Command Line Interface](#)
 - [Configuración de la AWS CLI](#)
2. Añada un perfil con nombre para el usuario administrador en el archivo config de la AWS CLI. Puede utilizar este perfil cuando ejecute los comandos de la AWS CLI. Para obtener más información sobre los perfiles con nombre, consulte [Perfiles con nombre](#) en la Guía del usuario de AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Para ver una lista de las regiones de AWS disponibles, consulte [Regiones y puntos de conexión](#) en la Referencia general de Amazon Web Services.

Note

El código y los comandos de ejemplo de este tutorial utilizan la región Oeste de EE. UU. (Oregón). Para usar una región diferente, cambie la región en el código y los comandos de este tutorial por la región que desea usar.

3. Verifique la configuración introduciendo el siguiente comando de ayuda en el símbolo del sistema:

```
aws help
```

Tras configurar una AWS cuenta y el AWS CLI, puede probar el siguiente ejercicio, en el que configurará una aplicación de ejemplo y probará la end-to-end configuración.

Paso siguiente

[Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink](#)

Paso 3: crear y ejecutar una aplicación de Managed Service para Apache Flink

En este ejercicio, deberá crear una aplicación de Managed Service para Apache Flink con flujos de datos como origen y receptor.

Esta sección contiene los siguientes pasos:

- [Crear dos Amazon Kinesis Data Streams](#)
- [Escribir registros de muestra en el flujo de entrada](#)
- [Descargar y examinar el código de Java de streaming de Apache Flink](#)
- [Compilar el código de la aplicación](#)
- [Cargar el código de Java de streaming de Apache Flink](#)
- [Crear y ejecutar la aplicación de Managed Service para Apache Flink](#)

Crear dos Amazon Kinesis Data Streams

Antes de crear una aplicación de Managed Service para Apache Flink para este ejercicio, cree dos flujos de datos de Kinesis (`ExampleInputStream` y `ExampleOutputStream`). Su aplicación utiliza estos flujos para los flujos de origen y destino de la aplicación.

Puede crear estos flujos mediante la consola de Amazon Kinesis o el siguiente comando de la AWS CLI. Para obtener instrucciones sobre la consola, consulte [Creación y actualización de flujos de datos](#) en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para crear flujos de datos (AWS CLI)

1. Para crear el primer flujo (`ExampleInputStream`), utilice el siguiente comando de la AWS CLI `create-stream` de Amazon Kinesis.

```
$ aws kinesis create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

2. Para crear el segundo flujo que la aplicación utilizará para escribir la salida, ejecute el mismo comando, cambiando el nombre a `ExampleOutputStream`.

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-west-2 \  
--profile adminuser
```

Escribir registros de muestra en el flujo de entrada

En esta sección, se utiliza un script de Python para escribir registros de muestra en el flujo para que la aplicación los procese.

Note

Esta sección requiere [AWS SDK for Python \(Boto\)](#).

1. Cree un archivo denominado `stock.py` con el siguiente contenido:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
            PartitionKey="partitionkey"
        )

if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Más adelante en el tutorial, se ejecuta el script `stock.py` para enviar datos a la aplicación.

```
$ python stock.py
```

Descargar y examinar el código de Java de streaming de Apache Flink

El código de la aplicación Java para este ejemplo está disponible en GitHub. Para descargar el código de la aplicación, haga lo siguiente:

1. Clone el repositorio remoto con el siguiente comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Vaya al directorio `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6`.

Tenga en cuenta lo siguiente en relación con el código de la aplicación:

- Un archivo [Project Object Model \(pom.xml\)](#) contiene información sobre la configuración y las dependencias de la aplicación, incluidas las bibliotecas de Managed Service para Apache Flink.
- El archivo `BasicStreamingJob.java` contiene el método `main` que define la funcionalidad de la aplicación.
- La aplicación utiliza un origen de Kinesis para leer datos del flujo de origen. El siguiente fragmento crea el origen de Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,  
        new SimpleStringSchema(), inputProperties));
```

- La aplicación crea conectores de origen y recepción para obtener acceso a recursos externos usando un objeto `StreamExecutionEnvironment`.
- La aplicación crea conectores de origen y recepción mediante propiedades estáticas. Para utilizar propiedades dinámicas de la aplicación, utilice los métodos `createSourceFromApplicationProperties` y `createSinkFromApplicationProperties` para crear los conectores. Estos métodos leen las propiedades de la aplicación para configurar los conectores.

Para obtener más información sobre las propiedades de tiempo de ejecución, consulte [Propiedades de tiempo de ejecución](#).

Compilar el código de la aplicación

En esta sección, se utiliza el compilador Apache Maven para crear el código de Java para la aplicación. Para obtener más información sobre la instalación de Apache Maven y el Java Development Kit (JDK), consulte [Requisitos previos para completar los ejercicios](#).

Note

Para utilizar el conector de Kinesis con versiones de Apache Flink anteriores a la 1.11, es necesario descargar el código fuente del conector y compilarlo tal y como se describe en la [documentación de Apache Flink](#).

Para compilar el código de la aplicación

1. Para utilizar el código de la aplicación, compile y empaquete el código en un archivo JAR. Puede compilar y empaquetar el código de una de las dos formas siguientes:
 - Utilice la herramienta de línea de comandos de Maven. Cree su archivo JAR ejecutando el siguiente comando en el directorio que contiene el archivo `pom.xml`:

```
mvn package
```

Note

El parámetro `-DFLink.version` no es necesario para la versión 1.0.1 de tiempo de ejecución de Managed Service para Apache Flink; solo es necesario para la versión 1.1.0 y versiones posteriores. Para obtener más información, consulte [the section called “Especificar la versión de Apache Flink de su aplicación”](#).

- Use el entorno de desarrollo. Consulte la documentación de su entorno de desarrollo para obtener más información.

Puede cargar el paquete como un archivo JAR o puede comprimir el paquete y cargarlo como un archivo ZIP. Si crea su aplicación mediante la AWS CLI, debe especificar el tipo de contenido del código (JAR o ZIP).

2. Si hay errores al compilar, verifique que la variable de entorno `JAVA_HOME` se ha configurado correctamente.

Si la aplicación se compila correctamente, se crea el siguiente archivo:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Cargar el código de Java de streaming de Apache Flink

En esta sección, creará un bucket de Amazon Simple Storage Service (Amazon S3) y cargará el código de la aplicación.

Para cargar el código de la aplicación

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija Crear bucket.
3. Escriba **ka-app-code-*<username>*** en el campo Nombre del bucket. Añada un sufijo al nombre del bucket, como su nombre de usuario, para que sea único a nivel global. Elija Siguiente.
4. En el paso Configurar opciones, deje los ajustes tal y como están y elija Siguiente.
5. En el paso Establecer permisos, deje los ajustes tal y como están y elija Siguiente.
6. Elija Crear bucket.
7. En la consola de Amazon S3, elija el *<username>*bucket ka-app-code- y, a continuación, seleccione Upload.
8. En el paso Seleccionar archivos, elija Añadir archivos. Vaya al archivo `aws-kinesis-analytics-java-apps-1.0.jar` que creó en el paso anterior. Elija Siguiente.
9. En el paso Establecer permisos, deje los ajustes tal y como están. Elija Siguiente.
10. En el paso Establecer propiedades, deje los ajustes tal y como están. Seleccione Cargar.

El código de la aplicación ya está almacenado en un bucket de Amazon S3 al que la aplicación puede acceder.

Crear y ejecutar la aplicación de Managed Service para Apache Flink

Puede crear y ejecutar una aplicación de Managed Service para Apache Flink mediante la consola o la AWS CLI.

Note

Cuando crea la aplicación mediante la consola, sus recursos AWS Identity and Access Management (de IAM) y de Amazon CloudWatch Logs se crean automáticamente. Si crea la aplicación mediante la AWS CLI, debe crear estos recursos por separado.

Temas

- [Crear y ejecutar la aplicación \(consola\)](#)
- [Crear y ejecutar la aplicación \(AWS CLI\)](#)

Crear y ejecutar la aplicación (consola)

Siga estos pasos para crear, configurar, actualizar y ejecutar la aplicación mediante la consola.

Crear la aplicación

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel de Managed Service para Apache Flink, seleccione Crear aplicación de análisis.
3. En la página Managed Service para Apache Flink: crear aplicación, proporcione los detalles de la aplicación de la siguiente manera:
 - En Nombre de la application, escriba **MyApplication**.
 - En Descripción, escriba **My java test app**.
 - En Tiempo de ejecución, escriba Apache Flink.

Note

Managed Service para Apache Flink utiliza la versión 1.8.2 o 1.6.2 de Apache Flink.

- Cambie el menú desplegable de versiones a Apache Flink 1.6.
4. Para los permisos de acceso, seleccione Crear o actualizar el rol de IAM. `kinesis-analytics-MyApplication-us-west-2`
 5. Elija Crear aplicación.

Note

Al crear una aplicación de Managed Service para Apache Flink mediante la consola, tiene la opción de tener un rol de IAM y una política creada para su aplicación. La aplicación utiliza este rol y la política para acceder a los recursos dependientes. Estos recursos de IAM reciben un nombre usando el nombre de la aplicación y la región tal y como se indica a continuación:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`

- Rol: `kinesisanalytics-MyApplication-us-west-2`

Editar la política de IAM

Edite la política de IAM para agregar permisos de acceso a los flujos de datos de Kinesis.

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Elija Políticas. Elija la política `kinesis-analytics-service-MyApplication-us-west-2` que la consola creó en su nombre en la sección anterior.
3. En la página Resumen, elija Editar política. Seleccione la pestaña JSON.
4. Añada la sección subrayada de la siguiente política de ejemplo a la política. Reemplace el ID de la cuenta de muestra (`012345678901`) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
```

```

    "Action": [
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
  },
  {
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Configurar la aplicación

1. En la MyApplication página, selecciona Configurar.
2. En la página Configurar aplicación, proporcione la Ubicación del código:
 - Para el bucket de Amazon S3, introduzca **ka-app-code-*<username>***.

- En Ruta al objeto de Amazon S3, introduzca **java-getting-started-1.0.jar**.
3. En Acceso a los recursos de la aplicación, en Permisos de acceso, seleccione Crear o actualizar el rol de IAM **kinesis-analytics-MyApplication-us-west-2**.
 4. Escriba las siguientes propiedades y valores de la aplicación:

ID de grupo	Clave	Valor
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

5. En Monitorización, asegúrese de que el Nivel de métricas de monitorización se ha establecido en Aplicación.
6. Para el CloudWatch registro, active la casilla Activar.
7. Seleccione Actualizar.

Note

Cuando eliges habilitar el CloudWatch registro de Amazon, Managed Service for Apache Flink crea un grupo de registros y un flujo de registros para ti. Los nombres de estos recursos son los siguientes:

- Grupo de registro: `/aws/kinesis-analytics/MyApplication`
- Flujo de registro: `kinesis-analytics-log-stream`

Ejecutar la aplicación

1. En la MyApplication página, selecciona Ejecutar. Confirme la acción.
2. Cuando la aplicación se está ejecutando, actualice la página. La consola muestra el Gráfico de la aplicación.

Detener la aplicación

En la MyApplication página, selecciona Detener. Confirme la acción.

Actualizar la aplicación

Mediante la consola, puede actualizar la configuración de la aplicación, tal como sus propiedades, ajustes de monitorización y la ubicación o el nombre de archivo JAR de la aplicación. También puede volver a cargar el JAR de la aplicación del bucket de Amazon S3 si necesita actualizar el código de la aplicación.

En la MyApplication página, elija Configurar. Actualice la configuración de la aplicación y elija Actualizar.

Crear y ejecutar la aplicación (AWS CLI)

En esta sección, se utiliza AWS CLI para crear y ejecutar la aplicación Managed Service para Apache Flink. Managed Service para Apache Flink utiliza el comando de la AWS CLI de `kinesisanalyticsv2` para crear e interactuar con las aplicaciones de Managed Service para Apache Flink.

Crear una política de permisos

En primer lugar, debe crear una política de permisos con dos instrucciones: una que concede permisos para la acción `read` en el flujo de origen y otra que concede permisos para las acciones `write` en el flujo de recepción. A continuación, asocie la política a un rol de IAM (que se crea en la siguiente sección). Por lo tanto, cuando Managed Service para Apache Flink asume el rol, el servicio tiene los permisos necesarios para leer desde el flujo de origen y escribir en el flujo de recepción.

Utilice el siguiente código para crear la política de permisos

`AKReadSourceStreamWriteSinkStream`. Reemplace *username* por el nombre de usuario que se utilizó para crear el bucket de Amazon S3 para almacenar el código de la aplicación. Sustituya el ID de la cuenta en los nombres de recurso de Amazon (ARN) (*012345678901*) por el ID de su cuenta.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
```

```

    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectVersion"
    ],
    "Resource": ["arn:aws:s3::ka-app-code-username",
      "arn:aws:s3::ka-app-code-username/*"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

Para step-by-step obtener instrucciones sobre cómo crear una política de permisos, consulte el [tutorial sobre cómo crear y adjuntar su primera política gestionada por el cliente](#) en la Guía del usuario de IAM.

Note

Para acceder a otros servicios de Amazon, puede usar AWS SDK for Java. Managed Service para Apache Flink establece automáticamente las credenciales requeridas por el SDK con las del rol de IAM de ejecución del servicio asociada a su aplicación. No hace falta realizar ningún otro paso.

Creación de un rol de IAM

En esta sección, creará un rol de IAM que la aplicación de Managed Service para Apache Flink pueda asumir para leer un flujo de origen y escribir en el flujo de recepción.

Managed Service para Apache Flink no puede acceder a su flujo sin permisos. Estos permisos se conceden a través del rol de IAM. Cada rol de IAM tiene dos políticas asociadas. La política de confianza concede a Managed Service para Apache Flink permiso para asumir el rol, y la política de permisos determina lo que Managed Service para Apache Flink puede hacer después de asumir el rol.

Usted deberá asociar la política de permisos que ha creado en la sección anterior a este rol.

Para crear un rol de IAM


1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Roles, Crear rol.
3. En Seleccionar tipo de entidad de confianza, elija Servicio de AWS. En Elegir el servicio que usará este rol, elija Kinesis. En Seleccionar su caso de uso, elija Kinesis Analytics.

Elija Siguiente: permisos.

4. En la página Asociar políticas de permisos, elija Siguiente: Revisión. Asociará políticas de permisos después de crear el rol.
5. En la página Crear rol, escriba **MF-stream-rw-role** como Nombre de rol. Elija Crear rol.

Ahora ha creado un nuevo rol de IAM llamado `MF-stream-rw-role`. A continuación, actualice las políticas de confianza y permisos del rol.

6. Asocie la política de permisos al rol.

 Note

Para este ejercicio, Managed Service para Apache Flink asume este rol tanto para leer datos de un flujo de datos de Kinesis (origen) como para escribir la salida en otro flujo de datos de Kinesis. Asocie la política que ha creado en el paso anterior, [the section called “Crear una política de permisos”](#).

- a. En la página Resumen, elija la pestaña Permisos.
- b. Seleccione Asociar políticas.
- c. En el campo de búsqueda, escriba **AKReadSourceStreamWriteSinkStream** (la política que ha creado en la sección anterior).
- d. Elija la ReadSourceStreamWriteSinkStream política AK y elija Adjuntar política.

Ahora ha creado el rol de ejecución de servicio que utiliza la aplicación para obtener acceso a los recursos. Anote el ARN del nuevo rol.

Para step-by-step obtener instrucciones sobre cómo crear un rol, consulte [Creación de un rol de IAM \(consola\)](#) en la Guía del usuario de IAM.

Crear la aplicación de Managed Service para Apache Flink

1. Copie el siguiente código JSON en un archivo denominado `create_request.json`. Cambie el ARN del rol de ejemplo por el ARN del rol que ha creado antes. Reemplace el sufijo del ARN del bucket (*username*) por el sufijo que eligió en la sección anterior. Reemplace el ID de la cuenta de ejemplo (*012345678901*) del rol de ejecución del servicio por el ID de su cuenta.

```
{
  "ApplicationName": "test",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_6",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "java-getting-started-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```

```
    }  
  ]  
}  
}
```

2. Ejecute la acción [CreateApplication](#) con la solicitud anterior para crear la aplicación:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://  
create_request.json
```

Se ha creado la aplicación. Puede iniciar la aplicación en el siguiente paso.

Iniciar la aplicación

En esta sección, se utiliza la acción [StartApplication](#) para iniciar la aplicación.

Para iniciar la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `start_request.json`.

```
{  
  "ApplicationName": "test",  
  "RunConfiguration": {  
    "ApplicationRestoreConfiguration": {  
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"  
    }  
  }  
}
```

2. Ejecute la acción [StartApplication](#) con la solicitud anterior para iniciar la aplicación:

```
aws kinesisanalyticstv2 start-application --cli-input-json file://start_request.json
```

Ya se debe estar ejecutando la aplicación. Puedes comprobar las métricas de Managed Service for Apache Flink en la CloudWatch consola de Amazon para comprobar que la aplicación funciona.

Detener la aplicación

En esta sección, se utiliza la acción [StopApplication](#) para detener la aplicación.

Para detener la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Ejecute la acción [StopApplication](#) con la siguiente solicitud para detener la aplicación:

```
aws kinesisanalyticstv2 stop-application --cli-input-json file://stop_request.json
```

La aplicación se habrá detenido.

Agrega una opción de CloudWatch registro

Puedes usar el AWS CLI para añadir un flujo de CloudWatch registro de Amazon a tu aplicación. Para obtener información sobre el uso CloudWatch de Logs con su aplicación, consulte [the section called "Configuración del registro"](#).

Actualizar propiedades de entorno

En esta sección, utilizará la acción [UpdateApplication](#) para cambiar las propiedades del entorno de la aplicación sin tener que volver a compilar el código de la aplicación. En este ejemplo, deberá cambiar la región de los flujos de origen y destino.

Para actualizar propiedades de entorno de la aplicación

1. Guarde el siguiente código JSON en un archivo denominado `update_properties_request.json`.

```
{"ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
```

```

        "AggregationEnabled" : "false"
      }
    },
    {
      "PropertyGroupId": "ConsumerConfigProperties",
      "PropertyMap" : {
        "aws.region" : "us-west-2"
      }
    }
  ]
}
}
}

```

2. Ejecute la acción [UpdateApplication](#) con la solicitud anterior para actualizar las propiedades del entorno:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Actualizar el código de la aplicación

Cuando necesite actualizar el código de la aplicación con una nueva versión del paquete de código, utilice la acción [UpdateApplication](#) de la AWS CLI.

Para usar la AWS CLI, elimine el paquete de código anterior de su bucket de Amazon S3, cargue la nueva versión y ejecute `UpdateApplication`, especificando el mismo nombre de bucket y objeto de Amazon S3 y la nueva versión del objeto. La aplicación se reiniciará con el nuevo paquete de código.

En el siguiente ejemplo de solicitud de la acción `UpdateApplication`, se vuelve a cargar el código de la aplicación y se reinicia la aplicación. Actualice la `CurrentApplicationVersionId` a la versión actual de la aplicación. Puede comprobar la versión actual de la aplicación mediante las acciones `ListApplications` o `DescribeApplication`. Actualice el sufijo del nombre del bucket (`<username>`) con el sufijo que haya elegido en la sección [the section called “Crear dos Amazon Kinesis Data Streams”](#).

```
{
  "ApplicationName": "test",
  "CurrentApplicationVersionId": 1,

```



```
"ApplicationConfigurationUpdate": {
  "ApplicationCodeConfigurationUpdate": {
    "CodeContentUpdate": {
      "S3ContentLocationUpdate": {
        "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
        "FileKeyUpdate": "java-getting-started-1.0.jar"
      }
    }
  }
}
```

Paso 4: Eliminar recursos de AWS

En esta sección se incluyen los procedimientos para eliminar los recursos de AWS creados en el tutorial de Introducción.

Este tema contiene las siguientes secciones:

- [Elimine su aplicación de Managed Service para Apache Flink](#)
- [Eliminar sus flujos de datos de Kinesis](#)
- [Eliminar el objeto y el bucket de Amazon S3](#)
- [Eliminar sus recursos de IAM](#)
- [CloudWatch Elimine sus recursos](#)

Elimine su aplicación de Managed Service para Apache Flink

1. Abra la consola de Kinesis en <https://console.aws.amazon.com/kinesis>.
2. En el panel Servicio gestionado para Apache Flink, elija MyApplication.
3. Elija Configurar.
4. En la sección Instantáneas, seleccione Deshabilitar y, a continuación, seleccione Actualizar.
5. En la página de la aplicación, seleccione Eliminar y, a continuación, confirme la eliminación.

Eliminar sus flujos de datos de Kinesis

1. Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
2. En el panel Kinesis Data Streams, ExampleInputStream elija.

3. En la ExampleInputStream página, elija Eliminar Kinesis Stream y, a continuación, confirme la eliminación.
4. En la página de transmisiones de Kinesis, elija, elija Acciones ExampleOutputStream, elija Eliminar y, a continuación, confirme la eliminación.

Eliminar el objeto y el bucket de Amazon S3

1. Abra la consola de Amazon S3 en <https://console.aws.amazon.com/s3/>.
2. Elija el cubo ka-app-code -. <username>
3. Elija Eliminar y luego ingrese el nombre del bucket para confirmar la eliminación.

Eliminar sus recursos de IAM

1. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En la barra de navegación, seleccione Políticas.
3. En el control de filtros, introduzca kinesis.
4. Elige la política kinesis-analytics-service- MyApplication -us-west-2.
5. Seleccione Acciones de política y, a continuación, Eliminar.
6. En la barra de navegación, seleccione Roles.
7. Elija la función kinesis-analytics - -us-west-2. MyApplication
8. Elija Eliminar rol y, a continuación, confirme la eliminación.

CloudWatch Elimine sus recursos

1. Abra la CloudWatch consola en <https://console.aws.amazon.com/cloudwatch/>.
2. En la barra de navegación, elija Registros.
3. Elija el grupo de registros MyApplication/aws/kinesis-analytics/.
4. Elija Eliminar grupo de registro y, a continuación, confirme la eliminación.

Configuración de Apache Flink

Managed Service para Apache Flink es una implementación del marco de Apache Flink. Managed Service para Apache Flink utiliza los valores predeterminados que se describen en esta sección. Algunos de estos valores se pueden establecer para las aplicaciones de Managed Service para Apache Flink en el código, y otros no se pueden cambiar.

Este tema contiene las siguientes secciones:

- [Configuración de Apache Flink](#)
- [Backend de estado](#)
- [Creación de puntos de control](#)
- [Usar punto de guardado](#)
- [Tamaños de montón](#)
- [Deblotear el búfer](#)
- [Propiedades de configuración de Flink modificables](#)
- [Visualización de las propiedades configuradas de Flink](#)

Configuración de Apache Flink

Managed Service para Apache Flink proporciona una configuración de Flink predeterminada que consta de los valores recomendados por Apache Flink para la mayoría de las propiedades y algunos basados en los perfiles de aplicación comunes. Para obtener más información sobre la configuración de Flink, consulte [Configuración](#). La configuración predeterminada proporcionada por el servicio funciona para la mayoría de las aplicaciones. Sin embargo, si necesita modificar las propiedades de configuración de Flink para mejorar el rendimiento de determinadas aplicaciones con un alto nivel de paralelismo y un uso elevado de memoria y estado, o si necesita habilitar nuevas características de depuración en Apache Flink, puede cambiar determinadas propiedades solicitando un caso de soporte. Para obtener más información, consulte el [Centro de soporte de AWS](#). Puede comprobar la configuración actual de su aplicación mediante el [panel de Apache Flink](#).

Backend de estado

Managed Service para Apache Flink almacena los datos transitorios en un backend de estado. Managed Service para Apache Flink usa RocksDBStateBackend. Llamar `setStateBackend` para configurar un backend diferente no tiene ningún efecto.

Habilitamos las siguientes características en el backend de estado:

- Instantáneas incrementales de backend de estado
- Instantáneas del backend de estado asíncrono
- Recuperación local de puntos de control

En Managed Service para Apache Flink, la `state.backend.rocksdb.ttl.compaction.filter.enabled` configuración está habilitada de forma predeterminada. Con este filtro, puede actualizar el código de la aplicación para activar la estrategia de limpieza y compactación. Para obtener más información, consulte [Tiempo de vida de estado \(TTL\) en Flink 1.8.0](#) en la [documentación de Apache Flink](#).

[Para obtener más información sobre los backends de estado, consulte Backends de estado en la documentación de Apache Flink.](#)

Creación de puntos de control

Managed Service para Apache Flink utiliza una configuración de puntos de control predeterminada con los siguientes valores. Algunos de estos valores se pueden cambiar. Debe establecer [CheckpointConfiguration.ConfigurationType](#) en CUSTOM para que Managed Service para Apache Flink utilice valores de puntos de control modificados.

Opción	¿Se puede modificar?	Cómo	Valor predeterminado
CheckpointingEnabled	Modificable	Crear aplicación Actualizar aplicación AWS CloudFormation	True
CheckpointInterval	Modificable	Crear aplicación Actualizar aplicación	60000

Opción	¿Se puede modificar?	Cómo	Valor predeterminado
		AWS CloudFormation	
MinPauseBetweenCheckpoints	Modificable	Crear aplicación Actualizar aplicación AWS CloudFormation	5000
Puntos de comprobación no alineados	Modificable	Caso de soporte	False
Número de puntos de control simultáneos	No se puede modificar	N/A	1
Modo de puntos de control	No se puede modificar	N/A	Exactamente una vez
Política de retención de puntos de control	No se puede modificar	N/A	Error
Tiempo de espera del punto de control	No se puede modificar	N/A	60 minutos
Número máximo de puntos de control retenidos	No se puede modificar	N/A	1
Estrategia de reinicio	No se puede modificar	N/A	Retraso fijo, con reintentos infinitos cada 10 segundos.
Ubicación del punto de control y del punto de guardado	No se puede modificar	N/A	Almacenamos datos duraderos de puntos de control y puntos de guardado en un bucket S3 propiedad del servicio.

Opción	¿Se puede modificar?	Cómo	Valor predeterminado
Umbral de memoria de backend de estado	No se puede modificar	N/A	1048576

Usar punto de guardado

De forma predeterminada, al restaurar desde un punto de guardado, la operación de reanudación intentará asignar todos los estados del punto guardado al programa con el que se está restaurando. Si ha eliminado un operador, de forma predeterminada, se producirá un error al restaurar desde un punto de guardado que contenga datos que correspondan al operador que falta. [Para permitir que la operación se realice correctamente, defina el parámetro `AllowNonRestoredState` de `FlinkRunConfiguration` de la aplicación en `true`](#). Esto permitirá que la operación de reanudación omita el estado que no se pueden asignar al nuevo programa.

Para obtener más información, consulte [Permitir estado no restaurado](#) en la [documentación de Apache Flink](#).

Tamaños de montón

Managed Service para Apache Flink asigna a cada KPU 3 GiB de montón de JVM y reserva 1 GiB para las asignaciones de código nativo. Para obtener información sobre cómo aumentar la capacidad de su aplicación, consulte [the section called “Escalado”](#).

Para obtener más información acerca del tamaño de los montones de JVM, consulte la [Configuración](#) en la [documentación de Apache Flink](#).

Deblotear el búfer

Deblotear el búfer puede ayudar a las aplicaciones con alta resistencia. Si su aplicación tiene puntos de control o puntos de guardado que fallan, activar esta característica podría resultar útil. Para ello, solicite un [caso de soporte](#).

Para obtener más información, consulte [El mecanismo de deblotear del búfer](#) en la documentación de [Apache Flink](#).

Propiedades de configuración de Flink modificables

A continuación se muestran los ajustes de configuración de Flink que puede modificar mediante un [caso de soporte](#). Puede modificar más de una propiedad a la vez y para varias aplicaciones al mismo tiempo especificando el prefijo de la aplicación. Si hay otras propiedades de configuración de Flink fuera de esta lista que desee modificar, especifique la propiedad exacta en su caso.

Tolerancia a errores

```
restart-strategy:
```

```
restart-strategy.fixed-delay.delay:
```

Puntos de control y backends de estado

```
state.backend:
```

```
state.backend.fs.memory-threshold:
```

```
state.backend.incremental:
```

Creación de puntos de control

```
execution.checkpointing.unaligned:
```

Métricas nativas de RocksDB

Las métricas nativas de RocksDB no se envían a CloudWatch. Una vez habilitadas, se puede acceder a estas métricas desde el panel de control de Flink o desde la API de REST de Flink con herramientas personalizadas.

Managed Service para Apache Flink permite a los clientes acceder a la última [API de REST](#) de Flink (o a la versión compatible que esté utilizando) en modo de solo lectura mediante la [API CreateApplicationPresignedURL](#). Esta API se utiliza en el propio panel de Flink, pero también se puede utilizar en herramientas de monitoreo personalizadas.

```
state.backend.rocksdb.compaction.style:
```

```
state.backend.rocksdb.memory.partitioned-index-filters:
```

```
state.backend.rocksdb.metrics.actual-delayed-write-rate:
```

state.backend.rocksdb.metrics.background-errors:
state.backend.rocksdb.metrics.block-cache-capacity:
state.backend.rocksdb.metrics.block-cache-pinned-usage:
state.backend.rocksdb.metrics.block-cache-usage:
state.backend.rocksdb.metrics.column-family-as-variable:
state.backend.rocksdb.metrics.compaction-pending:
state.backend.rocksdb.metrics.cur-size-active-mem-table:
state.backend.rocksdb.metrics.cur-size-all-mem-tables:
state.backend.rocksdb.metrics.estimate-live-data-size:
state.backend.rocksdb.metrics.estimate-num-keys:
state.backend.rocksdb.metrics.estimate-pending-compaction-bytes:
state.backend.rocksdb.metrics.estimate-table-readers-mem:
state.backend.rocksdb.metrics.is-write-stopped:
state.backend.rocksdb.metrics.mem-table-flush-pending:
state.backend.rocksdb.metrics.num-deletes-active-mem-table:
state.backend.rocksdb.metrics.num-deletes-imm-mem-tables:
state.backend.rocksdb.metrics.num-entries-active-mem-table:
state.backend.rocksdb.metrics.num-entries-imm-mem-tables:
state.backend.rocksdb.metrics.num-immutable-mem-table:
state.backend.rocksdb.metrics.num-live-versions:
state.backend.rocksdb.metrics.num-running-compactions:
state.backend.rocksdb.metrics.num-running-flushes:

`state.backend.rocksdb.metrics.num-snapshots:`

`state.backend.rocksdb.metrics.size-all-mem-tables:`

`state.backend.rocksdb.thread.num:`

Opciones avanzadas de backends de estado

`state.storage.fs.memory-threshold:`

Opciones completas de TaskManager

`task.cancellation.timeout:`

`taskmanager.jvm-exit-on-oom:`

`taskmanager.numberOfTaskSlots:`

`taskmanager.slot.timeout:`

`taskmanager.network.memory.fraction:`

`taskmanager.network.memory.max:`

`taskmanager.network.request-backoff.initial:`

`taskmanager.network.request-backoff.max:`

`taskmanager.network.memory.buffer-debloat.enabled:`

`taskmanager.network.memory.buffer-debloat.period:`

`taskmanager.network.memory.buffer-debloat.samples:`

`taskmanager.network.memory.buffer-debloat.threshold-percentages:`

Configuración de memoria

`taskmanager.memory.jvm-metaspace.size:`

`taskmanager.memory.jvm-overhead.fraction:`

`taskmanager.memory.jvm-overhead.max:`

`taskmanager.memory.managed.consumer-weights:`

`taskmanager.memory.managed.fraction:`

`taskmanager.memory.network.fraction:`

`taskmanager.memory.network.max:`

`taskmanager.memory.segment-size:`

`taskmanager.memory.task.off-heap.size:`

RPC/Akka

`akka.ask.timeout:`

`akka.client.timeout:`

`akka.framesize:`

`akka.lookup.timeout:`

`akka.tcp.timeout:`

Cliente

`client.timeout:`

Opciones avanzadas de clústeres

`cluster.intercept-user-system-exit:`

`cluster.processes.halt-on-fatal-error:`

Configuraciones del sistema de archivos

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

Opciones avanzadas de tolerancia a errores

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

Configuración de memoria

`jobmanager.memory.heap.size:`

Métricas

`metrics.latency.interval:`

Opciones avanzadas para el cliente y el punto de conexión REST

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Opciones de seguridad SSL avanzadas

`security.ssl.internal.handshake-timeout:`

Opciones avanzadas de programación

`slot.request.timeout:`

Opciones avanzadas para la interfaz de usuario web de Flink

`web.timeout:`

Visualización de las propiedades configuradas de Flink

Puede ver las propiedades de Apache Flink que haya configurado usted mismo o que haya solicitado que se modifiquen mediante un [caso de soporte](#) a través del Panel de Apache Flink y siguiendo estos pasos:

1. Vaya al Panel de Flink
2. En el panel de navegación izquierdo, seleccione Administrador de trabajos.

3. Seleccione Configuración para ver la lista de propiedades de Flink.

Configuración de Managed Service para Apache Flink para acceder a los recursos de una Amazon VPC

Puede configurar una aplicación de Managed Service para Apache Flink para conectarse a subredes privadas en una nube privada virtual (VPC) de su cuenta. Utilice Amazon Virtual Private Cloud (Amazon VPC) para crear una red privada para recursos como bases de datos, instancias de caché o servicios internos. Conecte su aplicación a la VPC para tener acceso a recursos privados durante su ejecución.

Este tema contiene las siguientes secciones:

- [Conceptos de Amazon VPC](#)
- [Permisos de aplicación de la VPC](#)
- [Acceso a Internet y a servicios para una aplicación de Managed Service para Apache Flink conectada a una VPC](#)
- [API de VPC de Managed Service para Apache Flink](#)
- [Ejemplo: uso de una VPC para acceder a los datos de un clúster de Amazon MSK](#)

Conceptos de Amazon VPC

Amazon VPC es la capa de red para Amazon EC2. Si es nuevo en Amazon EC2, consulte [¿Qué es Amazon EC2?](#) en la Guía del usuario de Amazon EC2 para instancias de Linux para obtener una breve información general.

A continuación, se indican los conceptos clave de las VPC:

- Una nube privada virtual (VPC) es una red virtual dedicada a su cuenta de AWS.
- Una subred es un rango de direcciones IP en su VPC.
- Una tabla de enrutamiento contiene un conjunto de reglas, denominadas rutas, que se utilizan para determinar hacia dónde se dirige el tráfico de red.
- Una puerta de enlace de Internet es un componente de la VPC escalado horizontalmente, redundante y de alta disponibilidad que permite la comunicación entre las instancias de su VPC e Internet. Por lo tanto, no plantea riesgos de disponibilidad ni restricciones de ancho de banda para el tráfico de red.

- Un punto de conexión de VPC le permite conectar de forma privada la VPC a los servicios admitidos de AWS y a servicios de punto de conexión de VPC basados en PrivateLink sin necesidad de una puerta de enlace de Internet, un dispositivo NAT, una conexión VPN o una conexión de AWS Direct Connect. Las instancias de su VPC no necesitan direcciones IP públicas para comunicarse con los recursos del servicio. El tráfico entre su VPC y el otro servicio no sale de la red de Amazon.

Para obtener más información sobre el servicio Amazon VPC, consulte la [Guía del usuario de Amazon Virtual Private Cloud](#).

Managed Service para Apache Flink crea [interfaces de red elásticas](#) en una de las subredes proporcionadas en la configuración de VPC de la aplicación. El número de interfaces de red elásticas creadas en las subredes de la VPC puede variar en función del paralelismo y el paralelismo por KPU de la aplicación. Para obtener más información sobre el escalado de aplicaciones, consulte [Escalado](#).

Note

Las configuraciones de VPC no son compatibles con las aplicaciones SQL.

Note

El servicio Managed Service para Apache Flink administra el estado de los puntos de control y las instantáneas de las aplicaciones que tienen una configuración de VPC.

Permisos de aplicación de la VPC

En esta sección se describen las políticas de permisos que la aplicación necesitará para funcionar con la VPC. Para obtener más información sobre el uso de políticas de permisos, consulte [Identidad y gestión de acceso para Amazon Managed Service para Apache Flink](#).

La siguiente política de permisos otorga a la aplicación los permisos necesarios para interactuar con una VPC. Para usar esta política de permisos, agréguela al rol de ejecución de su aplicación.

Política de permisos para acceder a una Amazon VPC

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeDhcpOptions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ENIReadWritePermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2>DeleteNetworkInterface"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

Al especificar los recursos de la aplicación mediante la consola (como CloudWatch Logs o una Amazon VPC), la consola modifica el rol de ejecución de la aplicación para conceder permiso de acceso a esos recursos. Solo necesita modificar manualmente el rol de ejecución de su aplicación si la crea sin utilizar la consola.

Acceso a Internet y a servicios para una aplicación de Managed Service para Apache Flink conectada a una VPC

Por defecto, al conectar una aplicación de Managed Service para Apache Flink a una VPC de su cuenta, esta no tiene acceso a Internet a menos que la VPC se lo proporcione. Si la aplicación necesita acceso a Internet, debe cumplirse lo siguiente:

- La aplicación de Managed Service para Apache Flink solo debe configurarse con subredes privadas.
- La VPC debe contener una puerta de enlace o instancia NAT en una subred pública.
- Debe existir una ruta para el tráfico saliente desde las subredes privadas a la puerta de enlace NAT en una subred pública.

Note

Varios servicios ofrecen [puntos de conexión de VPC](#). Puede utilizar puntos de conexión de VPC para conectarse a servicios de Amazon desde el interior de una VPC sin acceso a Internet.

El hecho de que una subred sea pública o privada depende de su tabla de enrutamiento. Cada tabla de enrutamiento tiene una ruta predeterminada que determina el siguiente salto para los paquetes que tienen un destino público.

- Para una subred privada: la ruta predeterminada apunta a una puerta de enlace NAT (nat-...) o a una instancia de NAT (eni-...).
- Para una subred pública: la ruta predeterminada apunta a una puerta de enlace de Internet (igw-...).

Una vez que haya configurado la VPC con una subred pública (con una NAT) y una o más subredes privadas, haga lo siguiente para identificar las subredes públicas y privadas:

- En el panel de navegación de la consola de la VPC, elija Subredes.
- Seleccione una subred y, a continuación, elija la pestaña Tabla de enrutamiento. Verifique la ruta predeterminada:

- Subred pública: Destino: 0.0.0.0/0, Objetivo: igw-...
- Subred privada: Destino: 0.0.0.0/0, Objetivo: nat-... o eni-...

Para asociar la aplicación Managed Service para Apache Flink con subredes privadas:

- Abra la consola de Managed Service para Apache Flink en <https://console.aws.amazon.com/flink>
- En la página de Aplicaciones de Managed Service para Apache Flink, elija su aplicación y, a continuación, seleccione Detalles de la aplicación.
- En la página de su aplicación, elija Configurar.
- En la sección Conectividad de VPC, elija la VPC que desea asociar a la aplicación. Elija las subredes y el grupo de seguridad asociados a la VPC que desee que utilice la aplicación para acceder a los recursos de la VPC.
- Elija Actualizar.

Información relacionada

[Creación de una VPC con subredes públicas y privadas](#)

[Conceptos básicos de la puerta de enlace NAT](#)

API de VPC de Managed Service para Apache Flink

Realice las siguientes operaciones de la API de Managed Service para Apache Flink a fin de gestionar las VPC de su aplicación. Para obtener información sobre el uso de la API de Managed Service para Apache Flink, consulte [Ejemplo de código de API](#).

CreateApplication

Utilice la acción [CreateApplication](#) para añadir una configuración de VPC a la aplicación durante su creación.

El siguiente código de solicitud de ejemplo para la acción CreateApplication incluye una configuración de VPC cuando se crea la aplicación:

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My-Application-Description",
```

```

"RuntimeEnvironment":"FLINK-1_15",
"ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
"ApplicationConfiguration": {
  "ApplicationCodeConfiguration":{
    "CodeContent":{
      "S3ContentLocation":{
        "BucketARN":"arn:aws:s3:::mybucket",
        "FileKey":"myflink.jar",
        "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    },
    "CodeContentType":"ZIPFILE"
  },
  "FlinkApplicationConfiguration":{
    "ParallelismConfiguration":{
      "ConfigurationType":"CUSTOM",
      "Parallelism":2,
      "ParallelismPerKPU":1,
      "AutoScalingEnabled":true
    }
  },
  "VpcConfigurations": [
    {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
    }
  ]
}
}

```

AddApplicationVpcConfiguration

Utilice la acción [AddApplicationVpcConfiguration](#) para añadir una configuración de VPC a la aplicación después de que esta ha sido creada.

El siguiente código de solicitud de ejemplo para la acción `AddApplicationVpcConfiguration` añade una configuración de VPC a una aplicación existente:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfiguration": {
    "SecurityGroupIds": [ "sg-0123456789abcdef0" ],

```

```
    "SubnetIds": [ "subnet-0123456789abcdef0" ]
  }
}
```

DeleteApplicationVpcConfiguration

Utilice la acción [DeleteApplicationVpcConfiguration](#) para eliminar una configuración de VPC de la aplicación.

El siguiente código de solicitud de ejemplo para la acción `AddApplicationVpcConfiguration` elimina una configuración de VPC existente de una aplicación:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfigurationId": "1.1"
}
```

UpdateApplication

Utilice la acción [UpdateApplication](#) para actualizar todas las configuraciones de VPC de una aplicación a la vez.

El siguiente código de solicitud de ejemplo para la acción `UpdateApplication` actualiza todas las configuraciones de VPC de una aplicación:

```
{
  "ApplicationConfigurationUpdate": {
    "VpcConfigurationUpdates": [
      {
        "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
        "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
        "VpcConfigurationId": "2.1"
      }
    ]
  },
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9
}
```

Ejemplo: uso de una VPC para acceder a los datos de un clúster de Amazon MSK

Para ver un tutorial completo sobre cómo acceder a los datos de un clúster de Amazon MSK en una VPC, consulte [Replicación MSK](#).

Resolución de problemas de Managed Service para Apache Flink

La información siguiente lo puede ayudar a solucionar los problemas que puedan presentarse con Amazon Managed Service para Apache Flink.

Temas

- [Solución de problemas de desarrollo](#)
- [Solución de problemas de tiempo de ejecución](#)

Solución de problemas de desarrollo

Temas

- [Gráficos de Apache Flink Flame](#)
- [Problema del proveedor de credenciales con el conector EFO 1.15.2](#)
- [Aplicaciones con conectores Kinesis no compatibles](#)
- [Error de compilación: "No se pudieron resolver las dependencias del proyecto"](#)
- [Opción no válida: "kinesisanalyticsv2"](#)
- [UpdateApplication La acción no es volver a cargar el código de la aplicación](#)
- [S3 StreamingFileSink FileNotFoundExceptions](#)
- [FlinkKafkaConsumer problema con stop with savepoint](#)
- [Bloqueo de Flink 1.15 Async Sink](#)
- [El procesamiento del origen de Amazon Kinesis Data Streams no funciona durante la partición nueva.](#)

Gráficos de Apache Flink Flame

Los gráficos de Flame están habilitados de forma predeterminada en las aplicaciones de las versiones compatibles de Managed Service para Apache Flink. Los gráficos de Flame pueden afectar el rendimiento de la aplicación si se mantiene el gráfico abierto, como se menciona en la [documentación de Flink](#).

Si desea deshabilitar los gráficos de Flame para la aplicación, genere un caso para solicitar que se los deshabilite para el ARN de la aplicación. Para obtener más información, consulte el [Centro de soporte de AWS](#).

Problema del proveedor de credenciales con el conector EFO 1.15.2

Existe un [problema conocido](#) con las versiones del conector EFO de Kinesis Data Streams anteriores a la 1.15.2, en donde `FlinkKinesisConsumer` no respeta la configuración del `CredentialProvider`. Debido a este problema, se descartan configuraciones válidas, lo que provoca que se utilice el proveedor de credenciales `AUTO`. Esto puede provocar un problema al utilizar el acceso entre cuentas a Kinesis mediante el conector EFO.

Para resolver este error, utilice la versión 1.15.3 o superior del conector EFO.

Aplicaciones con conectores Kinesis no compatibles

La versión 1.15 de Managed Service para Apache Flink [rechazará automáticamente el inicio o la actualización de las aplicaciones](#) si utilizan versiones de Kinesis Connector (anteriores a la versión 1.15.2) incluidas en archivos JAR o ZIP de las aplicaciones.

Error de rechazo

Aparecerá el siguiente error al enviar las llamadas de creación o actualización de la aplicación:

```
An error occurred (InvalidArgumentException) when calling the CreateApplication operation: An unsupported Kinesis connector version has been detected in the application. Please update flink-connector-kinesis to any version equal to or newer than 1.15.2.
For more information refer to connector fix: https://issues.apache.org/jira/browse/FLINK-23528
```

Pasos para solucionarlo

- Actualice la dependencia de la aplicación en `flink-connector-kinesis`. Si utiliza Maven como la herramienta de creación de su proyecto, siga [Actualice una dependencia de Maven](#). Si usa Gradle, siga [Actualice una dependencia de Gradle](#).
- Vuelva a empaquetar la aplicación.
- Cargue en un bucket de Amazon S3.
- Vuelva a enviar la solicitud de creación o actualización de la aplicación con la aplicación modificada que se acaba de cargar en el bucket de Amazon S3.

- Si sigue apareciendo el mismo mensaje de error, vuelva a comprobar las dependencias de la aplicación. Si el problema persiste, cree una incidencia de soporte técnico.

Actualice una dependencia de Maven

1. Abra el `pom.xml` del proyecto.
2. Encuentre las dependencias del proyecto. Aparecen así:

```
<project>

  ...

  <dependencies>

    ...

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
    </dependency>

    ...

  </dependencies>

  ...

</project>
```

3. Actualice `flink-connector-kinesis` a una versión igual o posterior a la 1.15.2. Por ejemplo:

```
<project>

  ...

  <dependencies>

    ...

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
```

```
        <version>1.15.2</version>
    </dependency>

    ...

</dependencies>

...

</project>
```

Actualice una dependencia de Gradle

1. Abra el `build.gradle` del proyecto (o `build.gradle.kts` para las aplicaciones de Kotlin).
2. Encuentre las dependencias del proyecto. Aparecen así:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. Actualice `flink-connector-kinesis` a una versión igual o posterior a la 1.15.2. Por ejemplo:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}
```



```
}  
  
...
```

Error de compilación: "No se pudieron resolver las dependencias del proyecto"

Para compilar las aplicaciones de ejemplo de Managed Service para Apache Flink, primero debe descargar y compilar el conector Apache Flink Kinesis y añadirlo a su repositorio local de Maven. Si el conector no se ha agregado a su repositorio, aparece un error de compilación similar al siguiente:

```
Could not resolve dependencies for project your project name: Failure to  
find org.apache.flink:flink-connector-kinesis_2.11:jar:1.8.2 in https://  
repo.maven.apache.org/maven2 was cached in the local repository, resolution will not be  
reattempted until the update interval of central has elapsed or updates are forced
```

Para resolver este error, debe descargar el código fuente de Apache Flink (versión 1.8.2 de <https://flink.apache.org/downloads.html>) correspondiente al conector. Para obtener instrucciones sobre cómo descargar, compilar e instalar el código fuente de Apache Flink, consulte [the section called "Uso del conector Apache Flink Kinesis Streams con versiones anteriores de Apache Flink"](#).

Opción no válida: "kinesisanalyticsv2"

Para usar la versión 2 de la API de Managed Service para Apache Flink, necesita la última versión de AWS Command Line Interface (AWS CLI).

Para obtener información sobre la actualización de AWS CLI, consulte [Instalar AWS Command Line Interface](#) en la Guía del usuario de AWS Command Line Interface.

UpdateApplication La acción no es volver a cargar el código de la aplicación

La [UpdateApplication](#) acción no volverá a cargar el código de la aplicación con el mismo nombre de archivo si no se especifica ninguna versión del objeto S3. Para volver a cargar el código de la aplicación con el mismo nombre de archivo, habilite el control de versiones en el bucket de S3 y especifique la versión del nuevo objeto mediante el parámetro `ObjectVersionUpdate`. Para obtener más información sobre la habilitación del control de versiones de objetos en un bucket de S3, consulte [Habilitación o deshabilitación del control de versiones](#).

S3 StreamingFileSink FileNotFoundExceptions

Las aplicaciones de Managed Service para Apache Flink pueden ejecutarse en un archivo `FileNotFoundException` parcial en curso cuando se parte de instantáneas si falta un archivo parcial en curso al que se hace referencia por su punto de guardado. Cuando se produce este modo de error, el estado de operador de la aplicación Managed Service para Apache Flink no suele ser recuperable y debe reiniciarse sin instantánea mediante `SKIP_RESTORE_FROM_SNAPSHOT`. Consulte el siguiente ejemplo de `stacktrace`:

```
java.io.FileNotFoundException: No such file or directory: s3://your-s3-bucket/pathj/
INSERT/2023/4/19/7/_part-2-1234_tmp_12345678-1234-1234-1234-123456789012
    at
    org.apache.hadoop.fs.s3a.S3AFileSystem.s3GetFileStatus(S3AFileSystem.java:2231)
    at
    org.apache.hadoop.fs.s3a.S3AFileSystem.innerGetFileStatus(S3AFileSystem.java:2149)
    at
    org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:2088)
    at org.apache.hadoop.fs.s3a.S3AFileSystem.open(S3AFileSystem.java:699)
    at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:950)
    at
    org.apache.flink.fs.s3hadoop.HadoopS3AccessHelper.getObject(HadoopS3AccessHelper.java:98)
    at
    org.apache.flink.fs.s3.common.writer.S3RecoverableMultipartUploadFactory.recoverInProgressPart
    ...
```

[Flink StreamingFileSink escribe registros en sistemas de archivos compatibles con la abstracción Flink. FileSystem](#) Dado que los flujos entrantes pueden ser ilimitados, los datos se organizan en archivos parciales de tamaño finito y se añaden nuevos archivos a medida que se escriben los datos. El ciclo de vida parcial y la política de renovación determinan el momento, el tamaño y la denominación de los archivos parciales.

Note

Para obtener más información, consulte [Ciclo de vida parcial](#).

Durante los puntos de control y los puntos de guardado (creación de instantáneas), se cambia el nombre de todos los archivos pendientes y se los confirma. Sin embargo, los archivos parciales en curso no se confirman, sino que se les cambia el nombre, y se guarda su referencia en los metadatos de los puntos de control o de guardado para utilizarlos cuando se restauren los trabajos. Con el

tiempo, estos archivos parciales en curso pasarán a estar Pendientes, se les cambiará el nombre y se los confirmará en un punto de control o guardado posterior.

A continuación se indican las causas principales y las medidas correctoras de la falta de un archivo parcial en curso:

- Instantánea obsoleta utilizada para iniciar la aplicación Managed Service for Apache Flink: solo la última instantánea del sistema tomada cuando se detiene o actualiza una aplicación se puede utilizar para iniciar una aplicación Managed Service for Apache Flink con Amazon S3. `StreamingFileSink` Para evitar este tipo de error, utilice la última instantánea del sistema.
- Esto ocurre, por ejemplo, cuando se selecciona una instantánea creada con `CreateSnapshot` en lugar de una instantánea activada por el sistema durante la detención o la actualización. El punto de guardado de la instantánea anterior guarda una out-of-date referencia a un archivo parcial en curso al que se le ha cambiado el nombre y al que se ha confirmado en un punto de control o guardado posterior.
- Esto también puede ocurrir cuando se selecciona una instantánea activada por el sistema de un evento de detención o actualización que no es el más reciente. Un ejemplo es una aplicación con la instantánea del sistema deshabilitada pero con configuración `RESTORE_FROM_LATEST_SNAPSHOT`. Por lo general, las aplicaciones de Managed Service for Apache Flink con Amazon S3 siempre `StreamingFileSink` deben tener habilitada y `RESTORE_FROM_LATEST_SNAPSHOT` configurada la instantánea del sistema.
- Se elimina el archivo parcial en curso: dado que el archivo parcial en curso se encuentra en un bucket de S3, pueden eliminarlo otros componentes o actores que tengan acceso al bucket.
 - Esto puede suceder si ha detenido la aplicación durante demasiado tiempo y la política de ciclo de vida de los buckets de [S3](#) ha eliminado el archivo de parte en curso al que hace referencia el punto de almacenamiento de la aplicación. `MultiPartUpload` Para evitar este tipo de errores, asegúrese de que la política de ciclo de vida de S3 Bucket MPU abarque un período lo suficientemente amplio para su caso de uso.
 - Esto también puede ocurrir cuando el archivo parcial en curso se elimina manualmente o mediante otro de los componentes del sistema. Para evitar este tipo de errores, asegúrese de que otros actores o componentes no eliminen los archivos parciales en curso.
- Condición de carrera en la que se activa un punto de control automático después del punto de guardado: esto afecta a las versiones de Managed Service para Apache Flink hasta la 1.13, inclusive. Este problema se ha corregido en la versión 1.15 de Managed Service para Apache Flink; migre su aplicación a la versión 1.15 de Managed Service para Apache Flink para evitar que se repita. También te sugerimos migrar de a. `StreamingFileSink` [FileSink](#)

- Cuando las aplicaciones se detienen o actualizan, Managed Service para Apache Flink activa un punto de guardado y detiene la aplicación en dos pasos. Si se activa un punto de control automático entre los dos pasos, el punto de guardado quedará inutilizable, ya que se cambiará el nombre del archivo parcial en curso y, posiblemente, se lo confirmará.

FlinkKafkaConsumer problema con stop with savepoint

Al utilizar la versión antigua, FlinkKafkaConsumer existe la posibilidad de que la aplicación se bloquee al actualizar, detener o escalar si tiene habilitadas las instantáneas del sistema. No hay ninguna solución publicada disponible para este [problema](#), por lo que le recomendamos que actualice a la nueva [KafkaSource](#) para mitigar este problema.

Si utiliza FlinkKafkaConsumer con las instantáneas habilitadas, existe la posibilidad de que, cuando el trabajo de Flink procese una detención con una solicitud de API de punto de guardado, FlinkKafkaConsumer falle con un error de tiempo de ejecución que indique `ClosedException`. En estas condiciones, la aplicación Flink se bloquea y se manifiesta como puntos de control fallidos.

Bloqueo de Flink 1.15 Async Sink

Existe un [problema conocido](#) con los AWS conectores de la AsyncSink interfaz de implementación de Apache Flink. Esto afecta a las aplicaciones que utilizan Flink 1.15 con los siguientes conectores:

- Para aplicaciones Java:
 - KinesisStreamsSink – `org.apache.flink:flink-connector-kinesis`
 - KinesisStreamsSink – `org.apache.flink:flink-connector-aws-kinesis-streams`
 - KinesisFirehoseSink – `org.apache.flink:flink-connector-aws-kinesis-firehose`
 - DynamoDbSink – `org.apache.flink:flink-connector-dynamodb`
- Aplicaciones SQL/TableAPI/Python de Flink:
 - kinesis – `org.apache.flink:flink-sql-connector-kinesis`
 - kinesis – `org.apache.flink:flink-sql-connector-aws-kinesis-streams`
 - firehose – `org.apache.flink:flink-sql-connector-aws-kinesis-firehose`
 - dynamodb – `org.apache.flink:flink-sql-connector-dynamodb`

Las aplicaciones afectadas experimentarán los siguientes síntomas:

- El trabajo de Flink está en estado de RUNNING, pero no está procesando datos.
- No se reinicia ningún trabajo.
- Se agota el tiempo de espera de los puntos de control.

El problema se debe a un [error](#) en el SDK de AWS que hace que no se muestren ciertos errores al solicitante cuando se utiliza el cliente de HTTP asíncrono. Esto hace que el receptor espere indefinidamente a que se complete una "solicitud en tránsito" durante una operación de vaciado de puntos de control.

Este problema se solucionó en el SDK de AWS a partir de la versión 2.20.144.

A continuación, se incluyen instrucciones sobre cómo actualizar los conectores afectados para usar la nueva versión de SDK de AWS en sus aplicaciones:

Temas

- [Actualice las aplicaciones Java](#)
- [Actualizar aplicaciones de Python](#)

Actualice las aplicaciones Java

Siga los procedimientos que se indican a continuación para actualizar las aplicaciones Java:

flink-connector-kinesis

Si la aplicación utiliza `flink-connector-kinesis`:

El conector de Kinesis usa el sombreado para empaquetar algunas dependencias, incluido el SDK de AWS, en el archivo jar del conector. Para actualizar la versión del SDK de AWS, realice el siguiente procedimiento para reemplazar estas clases sombreadas:

Maven

1. Agregue el conector de Kinesis y los módulos del SDK de AWS necesarios como dependencias del proyecto.
2. Configure `maven-shade-plugin`:
 - a. Añada un filtro para excluir las clases de SDK de AWS sombreadas al copiar el contenido del archivo jar del conector de Kinesis.

- b. Añada una regla de reubicación para mover las clases de SDK de AWS actualizadas al paquete, según lo previsto por el conector de Kinesis.

pom.xml

```
<project>
  ...
  <dependencies>
    ...
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
      <version>1.15.4</version>
    </dependency>

    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>kinesis</artifactId>
      <version>2.20.144</version>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>netty-nio-client</artifactId>
      <version>2.20.144</version>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>sts</artifactId>
      <version>2.20.144</version>
    </dependency>
    ...
  </dependencies>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>3.1.1</version>
        <executions>
```

```

    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        ...
        <filters>
          ...
          <filter>
            <artifact>org.apache.flink:flink-connector-
kinesis</artifact>
            <excludes>
              <exclude>org/apache/flink/kinesis/
shaded/software/amazon/awssdk/**</exclude>
              <exclude>org/apache/flink/kinesis/
shaded/org/reactivestreams/**</exclude>
              <exclude>org/apache/flink/kinesis/
shaded/io/netty/**</exclude>
              <exclude>org/apache/flink/kinesis/
shaded/com/typesafe/netty/**</exclude>
            </excludes>
          </filter>
          ...
        </filters>
        <relocations>
          ...
          <relocation>
            <pattern>software.amazon.awssdk</pattern>

            <shadedPattern>org.apache.flink.kinesis.shaded.software.amazon.awssdk</
shadedPattern>
          </relocation>
          <relocation>
            <pattern>org.reactivestreams</pattern>

            <shadedPattern>org.apache.flink.kinesis.shaded.org.reactivestreams</
shadedPattern>
          </relocation>
          <relocation>
            <pattern>io.netty</pattern>

            <shadedPattern>org.apache.flink.kinesis.shaded.io.netty</shadedPattern>
          </relocation>

```

```

                <relocation>
                    <pattern>com.typesafe.netty</pattern>

    <shadedPattern>org.apache.flink.kinesis.shaded.com.typesafe.netty</
shadedPattern>

                </relocation>
                ...
            </relocations>
            ...
        </configuration>
    </execution>
</executions>
</plugin>
    ...
</plugins>
    ...
</build>
</project>

```

Gradle

1. Agregue el conector de Kinesis y los módulos del SDK de AWS necesarios como dependencias del proyecto.
2. Ajuste la configuración de ShadowJar:
 - a. Excluya las clases de SDK de AWS sombreadas al copiar el contenido del archivo jar del conector de Kinesis.
 - b. Reubique las clases de SDK de AWS actualizadas al paquete previsto por el conector de Kinesis.

build.gradle

```

...
dependencies {
    ...
    flinkShadowJar("org.apache.flink:flink-connector-kinesis:1.15.4")

    flinkShadowJar("software.amazon.awssdk:kinesis:2.20.144")
    flinkShadowJar("software.amazon.awssdk:sts:2.20.144")
    flinkShadowJar("software.amazon.awssdk:netty-nio-client:2.20.144")
}

```



```
    ...
}
...
shadowJar {
    configurations = [project.configurations.flinkShadowJar]

    exclude("org/apache/flink/kinesis/shaded/software/amazon/awssdk/**/*.*.class")
    exclude("org/apache/flink/kinesis/shaded/org/reactivestreams/**/*.*.class")
    exclude("org/apache/flink/kinesis/shaded/io/netty/**/*.*.class")
    exclude("org/apache/flink/kinesis/shaded/com/typesafe/netty/**/*.*.class")

    relocate("software.amazon.awssdk",
"org.apache.flink.kinesis.shaded.software.amazon.awssdk")
    relocate("org.reactivestreams",
"org.apache.flink.kinesis.shaded.org.reactivestreams")
    relocate("io.netty", "org.apache.flink.kinesis.shaded.io.netty")
    relocate("com.typesafe.netty",
"org.apache.flink.kinesis.shaded.com.typesafe.netty")
}
...
```

Otros conectores afectados

Si la aplicación utiliza otro conector afectado:

Para actualizar la versión del SDK de AWS, se debe aplicar la versión del SDK en la configuración de compilación del proyecto.

Maven

Agregue la lista de materiales (BOM) del SDK de AWS a la sección de administración de dependencias del archivo `pom.xml` para aplicar la versión del SDK del proyecto.

`pom.xml`

```
<project>
  ...
  <dependencyManagement>
    <dependencies>
      ...
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
```

```
        <artifactId>bom</artifactId>
        <version>2.20.144</version>
        <scope>import</scope>
        <type>pom</type>
    </dependency>
    ...
</dependencies>
</dependencyManagement>
...
</project>
```

Gradle

Agregue la dependencia de la plataforma en la lista de materiales (BOM) del SDK de AWS para aplicar la versión del SDK del proyecto. Esto requiere Gradle 5.0 o una versión más reciente:

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

Actualizar aplicaciones de Python

Las aplicaciones de Python pueden usar conectores de dos maneras diferentes: empaquetando conectores y otras dependencias de Java como parte de un único uber-jar, o usar el archivo jar del conector directamente. Para corregir las aplicaciones afectadas por el bloqueo de Async Sink:

- Si la aplicación usa un uber jar, siga las instrucciones de [Actualice las aplicaciones Java](#).
- Para reconstruir los archivos jar del conector desde el origen, siga estos pasos:

Construcción de conectores desde el origen:

Requisitos previos, similares a los [requisitos de construcción](#) de Flink:

- Java 11

- Maven 3.2.5

flink-sql-connector-kinesis

1. Descargue el código fuente de Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Descomprima el código fuente:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navegue hasta el directorio de conectores de Kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. Compile e instale el archivo jar del conector e indique la versión del SDK de AWS requerida. Para acelerar la construcción, utilice `-DskipTests` para omitir la ejecución de las pruebas y `-Dfast` omitir las comprobaciones adicionales del código fuente:

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. Navegue hasta el directorio de conectores de Kinesis

```
cd ../flink-sql-connector-kinesis
```

6. Compile e instale el archivo jar del conector sql:

```
mvn clean install -DskipTests -Dfast
```

7. El jar resultante estará disponible en:

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-kinesis-streams

1. Descargue el código fuente de Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Descomprima el código fuente:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navegue hasta el directorio de conectores de Kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. Compile e instale el archivo jar del conector e indique la versión del SDK de AWS requerida. Para acelerar la construcción, utilice `-DskipTests` para omitir la ejecución de las pruebas y `-Dfast` omitir las comprobaciones adicionales del código fuente:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Navegue hasta el directorio de conectores de Kinesis

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. Compile e instale el archivo jar del conector sql:

```
mvn clean install -DskipTests -Dfast
```

7. El jar resultante estará disponible en:

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-kinesis-manguera de incendios

1. Descargue el código fuente de Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Descomprima el código fuente:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navegue hasta el directorio de conectores

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. Compile e instale el archivo jar del conector e indique la versión del SDK de AWS requerida. Para acelerar la construcción, utilice `-DskipTests` para omitir la ejecución de las pruebas y `-Dfast` omitir las comprobaciones adicionales del código fuente:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Navegue hasta el directorio de conectores sql

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. Compile e instale el archivo jar del conector sql:

```
mvn clean install -DskipTests -Dfast
```

7. El jar resultante estará disponible en:

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. Descargue el código fuente de Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-connector-aws-3.0.0/flink-connector-aws-3.0.0-src.tgz
```

2. Descomprima el código fuente:

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. Navegue hasta el directorio de conectores

```
cd flink-connector-aws-3.0.0
```

4. Compile e instale el archivo jar del conector e indique la versión del SDK de AWS requerida. Para acelerar la construcción, utilice `-DskipTests` para omitir la ejecución de las pruebas y `-Dfast` omitir las comprobaciones adicionales del código fuente:

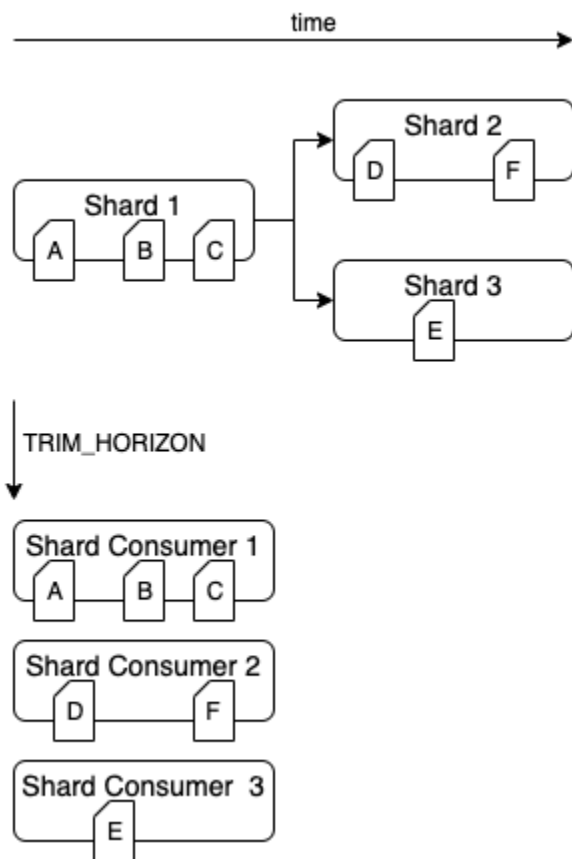
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -  
Daws.sdk.version=2.20.144
```

5. El jar resultante estará disponible en:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

El procesamiento del origen de Amazon Kinesis Data Streams no funciona durante la partición nueva.

La FlinkKinesisConsumer implementación actual no ofrece garantías sólidas de ordenamiento entre las particiones de Kinesis. Esto puede provocar que se procese durante la refragmentación de Kinesis Stream, en particular en el caso de las aplicaciones Flink que sufren retrasos en el out-of-order procesamiento. En algunas circunstancias, por ejemplo, los operadores de Windows basados en la hora de los eventos, es posible que los eventos se desechen debido a la demora resultante.



Este es un [problema conocido](#) en Open Source Flink. Hasta que se cuente con una corrección de conector, asegúrese de que sus aplicaciones de Flink no se queden atrás de las de Kinesis Data Streams al realizar la partición de nuevo. Al asegurarse de que sus aplicaciones de Flink toleran el retraso en el procesamiento, puede minimizar el impacto del out-of-order procesamiento y el riesgo de pérdida de datos.

Solución de problemas de tiempo de ejecución

Esta sección contiene información sobre el diagnóstico y la solución de problemas de tiempo de ejecución con su aplicación Managed Service para Apache Flink.

Temas

- [Herramientas para solucionar problemas](#)
- [Problemas con las aplicaciones](#)
- [La aplicación se está reiniciando](#)
- [El rendimiento es demasiado lento](#)
- [Crecimiento de estado ilimitado](#)
- [Operadores vinculados a E/S](#)
- [Limitación ascendente o de origen desde un flujo de datos de Kinesis](#)
- [Puntos de control](#)
- [Se está agotando el tiempo para llegar al punto de control](#)
- [Fallo en el punto de control de la aplicación Apache Beam](#)
- [Resistencia](#)
- [Sesgo de datos](#)
- [Sesgo de estado](#)
- [Integración con recursos en diferentes regiones](#)

Herramientas para solucionar problemas

La herramienta principal para detectar problemas en las aplicaciones son las alarmas de CloudWatch. Con las alarmas de CloudWatch, puede establecer umbrales para las métricas de CloudWatch que indiquen las condiciones de error o cuello de botella en su aplicación. Para obtener información sobre las alarmas de CloudWatch, consulte [Uso de CloudWatch alarmas con Amazon Managed Service para Apache Flink](#).

Problemas con las aplicaciones

Esta sección contiene soluciones para las condiciones de error que puede encontrar en su aplicación Managed Service para Apache Flink.

Temas

- [La aplicación está atascada en un estado transitorio](#)
- [La creación de la instantánea falla](#)
- [No puede obtener acceso a los recursos de una VPC](#)
- [Los datos se pierden al escribir en un bucket de Amazon S3](#)
- [La aplicación está en estado EJECUCIÓN pero no está procesando datos](#)
- [Error de instantánea, actualización de la aplicación o detención de la aplicación: `InvalidApplicationConfigurationException`](#)
- [`java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts`](#)

La aplicación está atascada en un estado transitorio

Si la aplicación permanece en un estado transitorio (STARTING, UPDATING, STOPPING, o AUTOSCALING), puede detenerla mediante la acción [StopApplication](#) con el `Force` parámetro establecido en `true`. No puede forzar la detención de una aplicación en el estado DELETING. Como alternativa, si la aplicación está en el estado UPDATING o AUTOSCALING, puede revertirla a la versión anterior en ejecución. Cuando se revierte una aplicación, se cargan los datos de estado de la última instantánea correcta. Si la aplicación no tiene instantáneas, Managed Service para Apache Flink rechaza la solicitud de reversión. [Para obtener más información sobre cómo revertir una aplicación, consulte acción `RollbackApplication`](#).

Note

La detención forzosa de la aplicación puede provocar la pérdida o la duplicación de datos. Para evitar la pérdida de datos o el procesamiento duplicado de los datos durante los reinicios de la aplicación, le recomendamos que tome instantáneas frecuentes de la aplicación.

Algunas causas para las aplicaciones atascadas incluyen las siguientes:

- El estado de la aplicación es demasiado grande: tener un estado de aplicación demasiado grande o demasiado persistente puede provocar que la aplicación se bloquee durante un punto de comprobación o una operación de captura de pantalla. Compruebe el `lastCheckpointDuration` de su aplicación y las métricas de `lastCheckpointSize` para valores que aumentan de forma constante o valores anormalmente altos.

- El código de la aplicación es demasiado grande: compruebe que el archivo JAR de la aplicación tenga un tamaño inferior a 512 MB. No se admiten archivos JAR de más de 512 MB.
- Falla la creación de la instantánea de la aplicación: Managed Service para Apache Flink toma una instantánea de la aplicación durante una solicitud de [UpdateApplication](#) o [StopApplication](#). Luego, el servicio utiliza este estado de instantánea y restaura la aplicación con la configuración de la aplicación actualizada para proporcionar una semántica de procesamiento exactamente una vez. Si falla la creación automática de instantáneas, consulte [La creación de la instantánea falla](#) siguiente.
- Falla la restauración a partir de una instantánea: si elimina o cambia un operador de una actualización de la aplicación e intenta realizar la restauración a partir de una instantánea, la restauración fallará de forma predeterminada si la instantánea contiene datos de estado del operador que falta. Además, la aplicación quedará bloqueada en el estado STOPPED o UPDATING. Para cambiar este comportamiento y permitir que la restauración se realice correctamente, cambie el parámetro allowNonRestoredState de [flinkRunConfiguration](#) de la aplicación a true. Esto permitirá que la operación de reanudación omita los datos de estado que no se pueden asignar al nuevo programa.
- La inicialización de la aplicación está tardando más tiempo: Managed Service para Apache Flink utiliza un tiempo de espera interno de 5 minutos (configuración predeterminada) mientras se espera a que se inicie un trabajo de Flink. Si su trabajo no se inicia dentro de este tiempo de espera, verá un registro de CloudWatch como se muestra a continuación:

```
Flink job did not start within a total timeout of 5 minutes for application: %s under account: %s
```

Si aparece el error anterior, significa que las operaciones definidas en el main método del trabajo de Flink están tardando más de 5 minutos, lo que provoca que se agote el tiempo de espera para la creación del trabajo de Flink en el extremo de Managed Service para Apache Flink. Le sugerimos que compruebe los registros del JobManager de Flink y el código de su aplicación para comprobar si se espera este retraso en main el método. De lo contrario, debe tomar medidas para solucionar el problema de modo que se complete en menos de 5 minutos.

Puede comprobar el estado de su aplicación mediante [ListApplications](#) o las acciones de [DescribeApplication](#).

La creación de la instantánea falla

El servicio Managed Service para Apache Flink no puede tomar una instantánea en las siguientes circunstancias:

- La aplicación ha superado el límite de instantáneas. El límite de instantáneas es 1000. Para obtener más información, consulte [Instantáneas](#).
- La aplicación no tiene permisos para acceder a su fuente o receptor.
- El código de la aplicación no funciona correctamente.
- La aplicación tiene otros problemas de configuración.

Si se produce una excepción al tomar una instantánea durante una actualización de la aplicación o al detener la aplicación, defina la `SnapshotsEnabled` propiedad de la aplicación [ApplicationSnapshotConfiguration](#) a `false` y vuelva a intentar la solicitud.

Las instantáneas pueden fallar si los operadores de la aplicación no están debidamente aprovisionados. Para obtener información sobre cómo ajustar el rendimiento de los operadores, consulte [Escalado de operadores](#)

Una vez que la aplicación vuelva a su estado correcto, se recomienda establecer la propiedad de `SnapshotsEnabled` de la aplicación en `true`.

No puede obtener acceso a los recursos de una VPC

Si la aplicación utiliza una VPC que se ejecuta en Amazon VPC, haga lo siguiente para comprobar que la aplicación tenga acceso a sus recursos:

- Compruebe los registros de CloudWatch para ver si aparece el siguiente error. Este error indica que la aplicación no puede acceder a los recursos de la VPC:

```
org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.
```

Si ve este error, compruebe que las tablas de enrutamiento estén configuradas correctamente y que los conectores tengan la configuración de conexión correcta.

Para obtener información acerca de la configuración y el análisis de los registros de CloudWatch, consulte [Registro y supervisión](#).

Los datos se pierden al escribir en un bucket de Amazon S3

Es posible que se produzcan algunas pérdidas de datos al escribir la salida en un bucket de Amazon S3 con Apache Flink versión 1.6.2. Recomendamos utilizar la última versión compatible de Apache Flink cuando utilice Amazon S3 para la salida directa. Para escribir en un bucket de Amazon S3 mediante Apache Flink 1.6.2, le recomendamos que utilice Kinesis Data Firehose. Para obtener más información sobre el uso de Kinesis Data Firehose con Managed Service para Apache Flink, consulte [Receptor de Kinesis Data Firehose](#)

La aplicación está en estado EJECUCIÓN pero no está procesando datos

Puede comprobar el estado de su aplicación mediante [ListApplications](#) o las acciones de [DescribeApplication](#). Si la aplicación introduce el estado RUNNING pero no escribe datos en el receptor, puede solucionar el problema añadiendo un flujo de registro de Amazon CloudWatch a la aplicación. Para obtener más información, consulte [Cómo trabajar con las opciones de CloudWatch registro de aplicaciones](#). El flujo de registro contiene mensajes que pueden ayudarlo a solucionar cualquier problema con la aplicación.

Error de instantánea, actualización de la aplicación o detención de la aplicación: InvalidApplicationConfigurationException

Puede producirse un error similar al siguiente durante una operación de instantánea o durante una operación que crea una instantánea, como la actualización o la detención de una aplicación:

```
An error occurred (InvalidApplicationConfigurationException) when calling the
UpdateApplication operation:
```

```
Failed to take snapshot for the application xxxx at this moment. The application is
currently experiencing downtime.
```

```
Please check the application's CloudWatch metrics or CloudWatch logs for any possible
errors and retry the request.
```

```
You can also retry the request after disabling the snapshots in the Managed Service for
Apache Flink console or by updating
the ApplicationSnapshotConfiguration through the AWS SDK
```

Este error se produce cuando la aplicación no puede crear una instantánea.

Si encuentra este error durante una operación de captura de pantalla o una operación que crea una instantánea, haga lo siguiente:

- Deshabilite las instantáneas para su aplicación. Puede hacerlo en la consola de Managed Service para Apache Flink o mediante el `SnapshotsEnabledUpdate` parámetro de la acción [UpdateApplication](#).
- Investigue por qué no se pueden crear instantáneas. Para obtener más información, consulte [La aplicación está atascada en un estado transitorio](#).
- Vuelva a activar las instantáneas cuando la aplicación vuelva a estar en buen estado.

`java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts`

La ubicación del almacén de confianza SSL se actualizó en una implementación anterior. Utilice los siguientes valores para el parámetro `ssl.truststore.location` en su lugar:

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

La aplicación se está reiniciando

Si la aplicación no está en buen estado, el trabajo en Apache Flink fallará y se reiniciará continuamente. En esta sección se describen los síntomas y los pasos para solucionar esta condición.

Síntomas

Esta condición puede tener los siguientes síntomas:

- La métrica de `FullRestarts` no es cero. Esta métrica representa el número de veces que se ha reiniciado el trabajo de la aplicación desde que se inició la aplicación.
- La métrica de `Downtime` no es cero. Esta métrica representa el número de milisegundos en los que la aplicación se encuentra en estado `FAILING` o `RESTARTING`.
- El registro de la aplicación contiene cambios de estado a `RESTARTING` o `FAILED`. Puede consultar estos cambios de estado en el registro de la aplicación mediante la siguiente consulta de CloudWatch Logs Insights: [Errores de análisis: errores relacionados con las tareas de la aplicación](#).

Causas y soluciones

Las siguientes condiciones pueden provocar que la aplicación se vuelva inestable y se reinicie repetidamente:

- El operador está emitiendo una excepción: si una excepción en un operador de la aplicación no se gestiona, la aplicación realiza una conmutación por error (interpretando que el operador no puede gestionar la falla). La aplicación se reinicia desde el último punto de control para mantener la semántica de procesamiento "una sola vez". Como resultado, Downtime no es cero durante estos períodos de reinicio. Para evitar que esto suceda, le recomendamos que gestione cualquier excepción reintentable en el código de la aplicación.

Puede investigar las causas de esta condición consultando los registros de la aplicación para ver si el estado de la aplicación cambió de RUNNING a FAILED. Para obtener más información, consulte [the section called “Errores de análisis: errores relacionados con las tareas de la aplicación”](#).

- Los flujos de datos de Kinesis no se suministran correctamente: si el origen o el receptor de la aplicación es un flujo de datos de Kinesis, compruebe si las [métricas](#) del flujo presentan errores ReadProvisionedThroughputExceeded o WriteProvisionedThroughputExceeded.

Si ve estos errores, puede aumentar el rendimiento disponible para el flujo de Kinesis aumentando el número de particiones del flujo. Para más información, consulte [¿Cómo puedo cambiar el número de particiones abiertas en Kinesis Data Streams?](#)

- No hay otros orígenes o receptores disponibles o no se los suministra correctamente: compruebe que la aplicación suministre correctamente los orígenes y los receptores. Verifique que se estén suministrando correctamente todos los orígenes o receptores utilizados en la aplicación (como otros servicios de AWS, u orígenes o destinos externos), que estos no presenten limitaciones de lectura o escritura o que no estén disponibles periódicamente.

Si tiene problemas de rendimiento con sus servicios dependientes, aumente los recursos disponibles para esos servicios o, bien, investigue la causa de cualquier error o falta de disponibilidad.

- No se suministran correctamente los operadores: si la carga de trabajo de los subprocesos de uno de los operadores de la aplicación no se distribuye correctamente, es posible que el operador se sobrecargue y la aplicación se bloquee. Para obtener información sobre cómo ajustar el paralelismo de los operadores, consulte [Gestionar correctamente el escalado del operador](#).
- La aplicación falla con DaemonException: este error aparece en el registro de la aplicación si utiliza una versión de Apache Flink anterior a la 1.11. Es posible que necesite actualizar a una versión posterior de Apache Flink para poder utilizar una versión de KPL 0.14 o posterior.
- La aplicación presenta una falla de TimeoutException, FlinkException o RemoteTransportException: estos errores pueden aparecer en el registro de la aplicación si los

administradores de tareas fallan. Si la aplicación está sobrecargada, los administradores de tareas pueden experimentar una presión en los recursos de CPU o memoria y hacer que fallen.

Estos errores pueden ser similares a los siguientes:

- `java.util.concurrent.TimeoutException: The heartbeat of JobManager with id xxx timed out`
- `org.apache.flink.util.FlinkException: The assigned slot xxx was removed`
- `org.apache.flink.runtime.io.network.netty.exception.RemoteTransportException: Connection unexpectedly closed by remote task manager`

Para solucionar este problema, compruebe lo siguiente:

- Compruebe las métricas de CloudWatch para ver si hay picos inusuales en el uso de la CPU o la memoria.
- Compruebe si la aplicación tiene problemas de rendimiento. Para obtener más información, consulte [Solución de problemas de desempeño](#).
- Examine el registro de la aplicación para detectar excepciones no gestionadas que esté generando el código de la aplicación.
- La aplicación falla y aparece el error `JaxbNotationModule Not Found`: este error se produce si la aplicación usa Apache Beam, pero no tiene las dependencias o versiones de dependencia correctas. Las aplicaciones de Managed Service para Apache Flink que utilizan Apache Beam deben utilizar las siguientes versiones de dependencias:

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
  <groupId>com.fasterxml.jackson.module</groupId>
  <artifactId>jackson-module-jaxb-annotations</artifactId>
  <version>2.10.2</version>
</dependency>
```

Si no proporciona la versión correcta de `jackson-module-jaxb-annotations` como dependencia explícita, la aplicación la cargará desde las dependencias del entorno y, como las versiones no coincidirán, la aplicación se bloqueará durante el tiempo de ejecución.

Para obtener más información sobre el uso de Apache Beam con Managed Service para Apache Flink, consulte [Uso de CloudFormation con Managed Service para Apache Flink](#).

- **La aplicación falla con `java.io.IOException: número insuficiente de búferes de red`**

La aplicación se está reiniciando

Esto ocurre cuando una aplicación no tiene suficiente memoria asignada para los búferes de red. Los búferes de red facilitan la comunicación entre las subtareas. Se los utiliza para almacenar registros antes de transmitirlos a través de una red y para almacenar los datos entrantes antes de dividirlos en registros y destinarlos a subtareas. La cantidad de búferes de red necesarios escala directamente según el paralelismo y la complejidad del gráfico de trabajo. Existen varios enfoques para mitigar este problema:

- Puede configurar un `parallelismPerKpu` más bajo para que haya más memoria asignada por subtask y búferes de red. Tenga en cuenta que bajar el `parallelismPerKpu` aumentará la KPU y, por lo tanto, el costo. Para evitarlo, puede reducir el paralelismo según el mismo factor para mantener la misma cantidad de KPU.
- Puede simplificar el gráfico de tareas reduciendo el número de operadores o encadenándolos para que se necesiten menos búferes.
- De lo contrario, puede ponerse en contacto con <https://aws.amazon.com/premiumsupport/> para obtener una configuración personalizada del búfer de red.

El rendimiento es demasiado lento

Si su aplicación no procesa los datos de streaming entrantes con la suficiente rapidez, tendrá un rendimiento deficiente y se volverá inestable. En esta sección se describen los síntomas y los pasos para solucionar esta condición.

Síntomas

Esta condición puede tener los siguientes síntomas:

- Si el origen de datos de su aplicación es un flujo de Kinesis, la `millisBehindLatest` métrica del flujo aumenta continuamente.
- Si el origen de datos de su aplicación es un clúster de Amazon MSK, las métricas de desfase de consumo del clúster aumentan continuamente. Para obtener más información, consulte [Monitoreo de desfase del consumidor](#) en la [Guía para desarrolladores de Amazon MSK](#).
- Si el origen de datos de su aplicación es un servicio o una fuente diferente, compruebe las métricas de desfase de consumo o los datos disponibles.

Causas y soluciones

La lentitud del rendimiento de las aplicaciones puede deberse a muchas causas. Si la aplicación no está al día con las entradas, compruebe lo siguiente:

- Si el desfase en el rendimiento aumenta repentinamente y luego se va reduciendo, compruebe si la aplicación se está reiniciando. La aplicación dejará de procesar las entradas mientras se reinicia, lo que provocará un aumento repentino del desfase. Para obtener más información acerca de las fallas de aplicaciones, consulte [La aplicación se está reiniciando](#).
- Si el desfase en el rendimiento es constante, compruebe si la aplicación está optimizada para el rendimiento. Para obtener información sobre cómo optimizar el rendimiento de la aplicación, consulte [Solución de problemas de desempeño](#).
- Si el desfase en el rendimiento no está aumentando repentinamente, sino que aumenta continuamente, y su aplicación está optimizada para el rendimiento, debe aumentar los recursos de la aplicación. Para obtener información sobre cómo aumentar los recursos de las aplicaciones, consulte [Escalado](#).
- Si su aplicación lee datos de un clúster de Kafka situado en una región diferente y `FlinkKafkaConsumer` o `KafkaSource` están prácticamente inactivos (con un nivel `idleTimeMsPerSecond` alto o `CPUUtilization` bajo) a pesar del elevado desfase del consumidor, puede aumentar el valor de `receive.buffer.byte`, por ejemplo, 2097152. Para obtener más información, consulte la sección sobre entornos de alta latencia en [Configuraciones personalizadas de MSK](#).

Para ver los pasos de solución de problemas relacionados con el rendimiento lento o el aumento del desfase del consumidor en el origen de la aplicación, consulte [Solución de problemas de desempeño](#).

Crecimiento de estado ilimitado

Si su aplicación no elimina adecuadamente la información de estado desactualizada, esta se acumulará continuamente y provocará problemas de rendimiento o estabilidad de la aplicación. En esta sección se describen los síntomas y los pasos para solucionar esta condición.

Síntomas

Esta condición puede tener los siguientes síntomas:

- La métrica `lastCheckpointDuration` está aumentando o teniendo picos gradualmente.

- La métrica `lastCheckpointSize` está aumentando o teniendo picos gradualmente.

Causas y soluciones

Las siguientes condiciones pueden provocar que su aplicación acumule datos de estado:

- Su aplicación retiene los datos de estado por más tiempo del necesario.
- Su aplicación utiliza consultas de ventana con una duración demasiado larga.
- No configuró el TTL para sus datos de estado. Para obtener más información, consulte [Tiempo de vida de estado \(TTL\)](#) en la [documentación de Apache Flink](#).
- Está ejecutando una aplicación que depende de la versión 2.25.0 o posterior de Apache Beam. Puede excluirse de la nueva versión de la transformación de lectura [ampliando sus `BeamApplicationProperties`](#) con los experimentos clave y el valor `use_deprecated_read`. Para obtener más información, consulte la [documentación de Apache Beam](#).

A veces, las aplicaciones se enfrentan a un crecimiento cada vez mayor del tamaño de estado, lo que no es sostenible a largo plazo (al fin y al cabo, una aplicación de Flink se ejecuta indefinidamente). A veces, esto se debe a que las aplicaciones almacenan los datos en estado y no conservan adecuadamente la información antigua. Pero a veces hay expectativas irrazonables sobre lo que Flink puede ofrecer. Las aplicaciones pueden usar agregaciones durante largos períodos de tiempo que abarcan días o incluso semanas. A menos que se utilice [AggregateFunctions](#), que permite agregaciones incrementales, Flink necesita mantener los eventos de toda la ventana en estado.

Además, al utilizar funciones de proceso para implementar operadores personalizados, la aplicación debe eliminar los datos del estado que ya no son necesarios para la lógica de negocios. En ese caso, el [tiempo de vida útil del estado](#) se puede utilizar para envejecer automáticamente los datos en función del tiempo de procesamiento. Managed Service para Apache Flink utiliza puntos de control incrementales y, por lo tanto, el ttl del estado se basa en la [compactación de RockSDB](#). Solo se puede observar una reducción real en el tamaño del estado (indicada por el tamaño del punto de control) después de una operación de compactación. En concreto, en el caso de los puntos de control con un tamaño inferior a 200 MB, es poco probable que se observe una reducción en el tamaño de los puntos de control como consecuencia de la caducidad del estado. Sin embargo, los puntos de guardado se basan en una copia limpia del estado que no contiene datos antiguos, por lo que puede activar una instantánea en Managed Service para Apache Flink para forzar la eliminación del estado obsoleto.

Con fines de depuración, puede resultar útil desactivar los puntos de control incrementales para comprobar con mayor rapidez si el tamaño de los puntos de control realmente disminuye o se estabiliza (y evitar así el efecto de compactación en RocksBS). Sin embargo, esto requiere una solicitud de asistencia con el equipo de servicio.

Operadores vinculados a E/S

Es mejor evitar las dependencias de sistemas externos en la ruta de datos. Suele ser mucho más eficaz mantener un conjunto de datos de referencia en estado en lugar de consultar un sistema externo para enriquecer los eventos individuales. Sin embargo, a veces hay dependencias que no se pueden cambiar fácilmente de estado, por ejemplo, si desea enriquecer los eventos con un modelo de machine learning alojado en Amazon Sagemaker.

Los operadores que interactúan con sistemas externos a través de la red pueden convertirse en un obstáculo y causar contrapresión. Se recomienda encarecidamente utilizar [AsynclO](#) para implementar la funcionalidad, reducir el tiempo de espera de las llamadas individuales y evitar que toda la aplicación se ralentice.

Además, para las aplicaciones con operadores vinculados a E/S, también puede tener sentido aumentar el ajuste [ParallelismPerKPU](#) de la aplicación Managed Service para Apache Flink. Este ajuste describe el número de subtareas paralelas que una aplicación puede realizar por unidad de procesamiento de Kinesis (KPU). Al aumentar el valor predeterminado de 1 a, por ejemplo, 4, la aplicación aprovecha los mismos recursos (y tiene el mismo costo), pero se puede escalar hasta cuadruplicar el paralelismo. Esto funciona bien para las aplicaciones vinculadas a la E/S, pero genera una sobrecarga adicional para las aplicaciones que no están vinculadas a la E/S.

Limitación ascendente o de origen desde un flujo de datos de Kinesis

Síntoma: la aplicación encuentra `LimitExceededExceptions` de su flujo de datos de Kinesis de origen ascendente.

Causa potencial: la configuración predeterminada del conector Kinesis de la biblioteca Apache Flink está configurada para leer desde la fuente de flujo de datos de Kinesis, con una configuración predeterminada muy agresiva para el número máximo de registros recuperados por llamada `GetRecords`. Apache Flink está configurado de forma predeterminada para recuperar 10.000 registros por llamada a `GetRecords` (esta llamada se realiza de forma predeterminada cada 200 ms), aunque el límite por partición es de solo 1000 registros.

Este comportamiento predeterminado puede provocar una limitación al intentar consumir datos del flujo de datos de Kinesis, lo que afectará al rendimiento y la estabilidad de las aplicaciones.

Esto se puede confirmar comprobando la `ReadProvisionedThroughputExceeded` métrica de CloudWatch y viendo períodos prolongados o sostenidos en los que esta métrica es superior a cero.

El cliente también podrá ver esto en los registros de CloudWatch de su aplicación Managed Service para Apache Flink al ver errores continuos `LimitExceededException`.

Solución: el cliente puede hacer una de estas dos cosas para resolver este escenario:

- Reducir el límite predeterminado de la cantidad de registros recuperados por llamada a `GetRecords`
- El cliente puede habilitar las lecturas adaptables en su aplicación Managed Service para Apache Flink. [Para obtener más información sobre la característica de lecturas adaptables, consulte `SHARD_USE_ADAPTIVE_READS`](#)

Puntos de control

Los puntos de control son el mecanismo de Flink para garantizar que el estado de una aplicación sea tolerante a errores. El mecanismo permite a Flink recuperar el estado de los operadores si el trabajo falla y proporciona a la aplicación la misma semántica que cuando se ejecuta sin errores. Con Managed Service para Apache Flink, el estado de una aplicación se almacena en RockSDB, un almacén de claves/valores integrado que mantiene su estado de funcionamiento en el disco. Cuando se toma un punto de control, el estado también se carga en Amazon S3, por lo que, incluso si se pierde el disco, el punto de control se puede utilizar para restaurar el estado de la aplicación.

Para obtener más información, consulte [¿Cómo funciona la creación de instantáneas de estado?](#) .

Fases de creación de punto de comprobación

La subtarea de un operador de puntos de control en Flink consta de 5 etapas principales:

- Espera [desfase de inicio]: Flink utiliza barreras de puntos de control que se insertan en el flujo, por lo que el tiempo en esta fase es el tiempo que el operador espera a que la barrera del punto de control llegue a ella.
- Alineación [Duración de la alineación]: en esta fase, la subtarea ha alcanzado una barrera, pero está esperando a que aparezcan barreras de otros flujos de entrada.
- Puntos de control de sincronización [Duración de la sincronización]: en esta fase, la subtarea toma la instantánea del estado del operador y bloquea el resto de las actividades de la subtarea.

- Puntos de control asíncronos [Duración asíncrona]: la mayor parte de esta fase consiste en que la subtarea carga el estado a Amazon S3. Durante esta fase, la subtarea ya no está bloqueada y puede procesar registros.
- Confirmación: normalmente se trata de una fase corta y consiste simplemente en la subtarea de enviar una confirmación al JobManager y también de ejecutar cualquier mensaje de confirmación (por ejemplo, con los receptores Kafka).

Cada una de estas fases (aparte de la Confirmación) se asigna a una métrica de duración de los puntos de comprobación que está disponible en la WebUI de Flink, que puede ayudar a aislar la causa del punto de control largo.

Para ver una definición exacta de cada una de las métricas disponibles en los puntos de control, vaya a la [pestaña Historial](#).

Investigación

Al investigar la duración prolongada de los puntos de control, lo más importante es determinar el cuello de botella del punto de control, es decir, qué operador y subtarea están tardando más en llegar al punto de control y qué fase de esa subtarea está tardando más tiempo. Esto se puede determinar mediante la WebUI de Flink en la tarea de punto de comprobación de trabajos. La interfaz web de Flink proporciona datos e información que ayudan a investigar los problemas relacionados con los puntos de control. Para obtener un desglose completo, consulte [Supervisión de los puntos de control](#).

Lo primero que hay que tener en cuenta es la Duración de principio a fin de cada operador en el gráfico de tareas para determinar qué operador está tardando en llegar al punto de comprobación y merece una investigación más profunda. Según la documentación de Flink, la definición de la duración es:

La duración desde la marca de tiempo de activación hasta la última confirmación (o n/a si aún no se ha recibido ninguna confirmación). La duración de un punto de control completo de principio a fin viene determinada por la última subtarea que confirma el punto de comprobación. Este tiempo suele ser superior al que necesitan las subtareas individuales para comprobar realmente el estado.

Las demás duraciones del punto de comprobación también proporcionan información más detallada sobre a qué se dedica el tiempo.

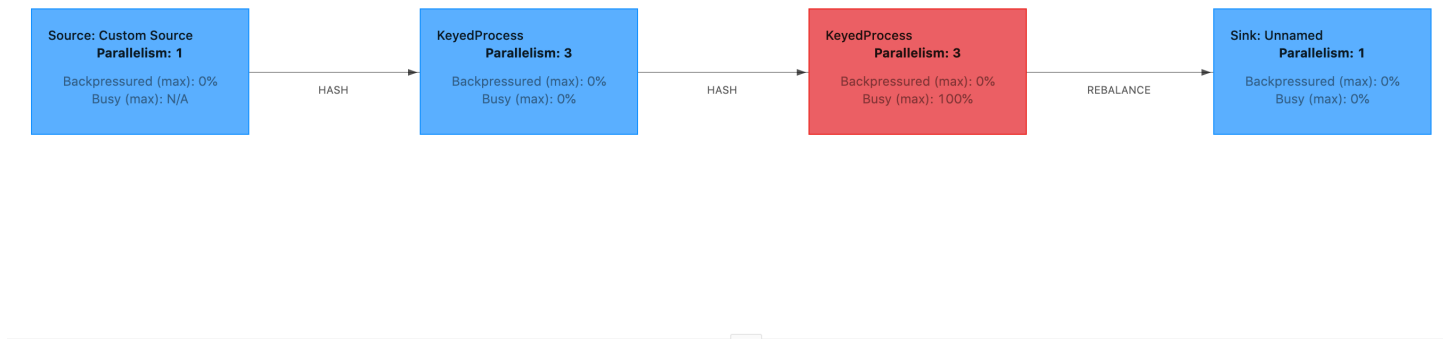
Si la Duración de la sincronización es alta, esto indica que algo está sucediendo durante la captura de pantalla. Durante esta fase se utiliza `snapshotState()` para clases que implementan la interfaz

SnapshotState; puede ser código de usuario, por lo que los volcados de subprocessos pueden resultar útiles para investigar este aspecto.

Una Duración asíncrona prolongada sugeriría que se está dedicando mucho tiempo a cargar el estado en Amazon S3. Esto puede ocurrir si el estado es grande o si se están cargando muchos archivos de estado. Si este es el caso, vale la pena investigar cómo utiliza la aplicación el estado y asegurarse de que se utilizan las estructuras de datos nativas de Flink siempre que sea posible ([con el estado clave](#)). Managed Service para Apache Flink configura Flink de tal manera que se minimiza el número de llamadas a Amazon S3 y se asegura de que no se prolongue demasiado. El siguiente es un ejemplo de las estadísticas de puntos de control de un operador. Muestra que la Duración asíncrona es relativamente larga en comparación con las estadísticas de puntos de control del operador anteriores.

SubTasks:									
	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay		
Minimum	495ms	11.1 KB	8ms	357ms	0 B (0 B)	0ms	126ms		
Average	813ms	586 KB	28ms	653ms	0 B (0 B)	0ms	126ms		
Maximum	1s	1.70 MB	69ms	1s	0 B (0 B)	1ms	128ms		
ID	Acknowledged	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay	Unaligned Checkpoint
0	2022-03-02 14:16:49	566ms	11.1 KB	8ms	429ms	0 B (0 B)	0ms	126ms	false
1	2022-03-02 14:16:50	1s	1.70 MB	69ms	1s	0 B (0 B)	0ms	128ms	false
2	2022-03-02 14:16:49	495ms	11.1 KB	8ms	357ms	0 B (0 B)	1ms	126ms	false
<div style="display: flex; justify-content: space-between; align-items: center;"> [-] Sink: Unnamed 1/1 (100%) 2022-03-02 14:16:49 131ms 0 B 0 B (0 B) </div>									
SubTasks:									

Si el Desfase de inicio es alto, indicaría que la mayor parte del tiempo se dedica a esperar a que la barrera del punto de control llegue al operador. Esto indica que la solicitud está tardando un tiempo en procesar los registros, lo que significa que la barrera está pasando lentamente por el gráfico de tareas. Este suele ser el caso si el Trabajo tiene resistencia o si un operador está constantemente ocupado. El siguiente es un ejemplo de un JobGraph en el que el segundo operador de KeyedProcess está ocupado.



Puede investigar qué está tardando tanto utilizando los gráficos de Flink Flame o los volcados de subprocesos de TaskManager. Una vez identificado el cuello de botella, se puede investigar más a fondo utilizando gráficos tipo llama o volcados de subprocesos.

Volcados de subprocesos

Los volcados de subprocesos son otra herramienta de depuración que se encuentra en un nivel ligeramente inferior al de los gráficos tipo llama. Un volcado de subprocesos genera el estado de ejecución de todos los subprocesos en un momento dado. Flink toma un volcado de subprocesos de JVM, que es un estado de ejecución de todos los subprocesos del proceso de Flink. El estado de un subproceso se presenta mediante un seguimiento de la pila del subproceso, así como información adicional. En realidad, los gráficos tipo llama se crean utilizando múltiples seguimientos de la pila tomados en rápida sucesión. El gráfico es una visualización hecha a partir de estos seguimientos que facilita la identificación de las rutas de código más comunes.

```
"KeyedProcess (1/3)#0" prio=5 Id=1423 RUNNABLE
  at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:154)
  at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>>19)
  at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
  at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperator
  at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStreamTask
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTask
```

```
at app//  
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces  
...
```

Arriba hay un fragmento de un volcado de subprocessos tomado de la interfaz de usuario de Flink para un solo subprocesso. La primera línea contiene información general sobre este subprocesso, que incluye:

- El nombre del subprocesso `KeyedProcess (1/3) #0`
- Prioridad del subprocesso `prio=5`
- Un identificador de subprocesso único `Id=1423`
- Estado del subprocesso: `EJECUTABLE`

El nombre de un subprocesso generalmente proporciona información sobre el propósito general del subprocesso. Los subprocessos del operador se pueden identificar por su nombre, ya que los subprocessos del operador tienen el mismo nombre que el operador, así como una indicación de la subtarea con la que están relacionados, por ejemplo, el subprocesso `KeyedProcess (1/3)#0` proviene del operador `KeyedProcess` y pertenece a la primera subtarea (de 3).

Los subprocessos pueden tener uno de los siguientes estados:

- **NUEVO**: el subprocesso se ha creado pero aún no se ha procesado
- **EJECUTABLE**: el subprocesso se está ejecutando en la CPU
- **BLOQUEADO**: el subprocesso está esperando a que otro subprocesso libere su bloqueo
- **ESPERANDO**: el subprocesso está en espera mediante un método `wait()`, `join()`, o `park()`
- **PROGRAMADO_ESPERANDO**: el subprocesso está en espera mediante el método `dormir`, `esperar`, `unirse` o `ubicar`, pero con un tiempo de espera máximo.

Note

En Flink 1.13, la profundidad máxima de un único `stacktrace` en el volcado de subprocessos está limitada a 8.

Note

Los volcados de subprocesos deberían ser el último recurso para depurar problemas de rendimiento en una aplicación de Flink, ya que pueden resultar difíciles de leer y requieren la toma de múltiples muestras y su análisis manual. Si es posible, es preferible utilizar gráficos tipo llama.

El subproceso se vuelca en Flink

En Flink, se puede realizar un volcado de subprocesos seleccionando la opción Administradores de tareas en la barra de navegación izquierda de la interfaz de usuario de Flink, seleccionando un administrador de tareas específico y, a continuación, navegando a la pestaña Volcado de subprocesos. El volcado de subprocesos se puede descargar, copiar a su editor de texto favorito (o analizador de volcado de subprocesos) o analizar directamente dentro de la vista de texto de la interfaz de usuario web de Flink (sin embargo, esta última opción puede resultar un poco torpe).

Para determinar qué Administrador de tareas tomar, se puede usar un volcado de subprocesos de la pestaña TaskManagers cuando se elige un operador en particular. Esto muestra que el operador se ejecuta diferentes subtareas de un operador y se puede ejecutar diferentes Administradores de tareas.

The screenshot displays the Flink web interface. On the left, a diagram shows a red box labeled 'KeyedProcess' with 'Parallelism: 3'. It has 'Backpressured (max): 0%' and 'Busy (max): 100%' below it. A dashed blue arrow labeled 'HASH' points to the box, and another labeled 'REBALANCE' points away from it. On the right, a table shows the 'TaskManagers' tab. The table has columns for Host, LOG, Bytes received, Records received, Bytes sent, Records sent, and Status. Two TaskManagers are listed, both in a 'RUNNING' state.

Host	LOG	Bytes received	Records received	Bytes sent	Records sent	Status
ip-142-151-131-22:61 21	LOG	936 B	0	0 B	0	RUNNING
ip-142-151-146-195:6 121	LOG	103 KB	1,423	71.1 KB	1,422	RUNNING

El volcado estará compuesto por múltiples seguimientos de pila. Sin embargo, cuando se investiga el volcado, lo más importante es lo que está relacionado con un operador. Se pueden encontrar fácilmente, ya que los subprocesos del operador tienen el mismo nombre que el operador, así como

una indicación de la subtarea con la que están relacionados. Por ejemplo, el siguiente seguimiento de la pila proviene del operador `KeyedProcess` y es la primera subtarea.

```
"KeyedProcess (1/3)#0" prio=5 Id=595 RUNNABLE
  at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:155)
  at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
  at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
  at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperat
  at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
  at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
  ...
```

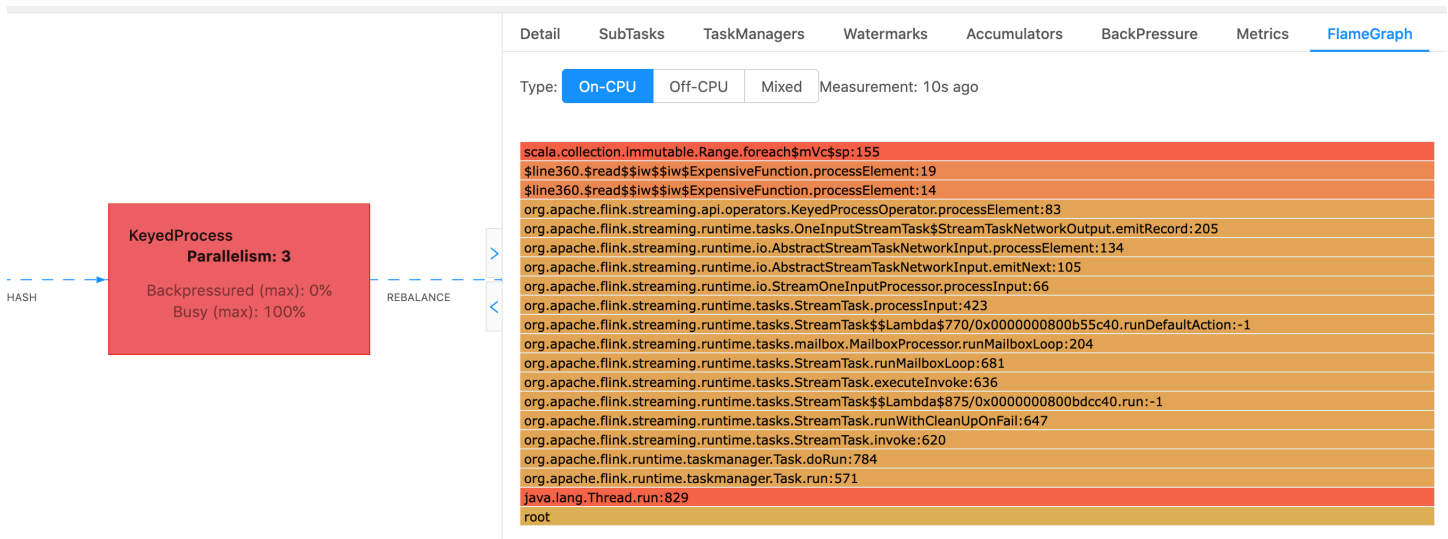
Esto puede resultar confuso si hay varios operadores con el mismo nombre, pero podemos ponerle nombre a los operadores para evitarlo. Por ejemplo:

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

Gráficos tipo llama

Los gráficos tipo llama son una útil herramienta de depuración que visualiza los seguimientos de pila del código objetivo, lo que permite identificar las rutas de código más frecuentes. Se crean muestreando los seguimientos de las pilas varias veces. El eje x de un gráfico tipo llama muestra los diferentes perfiles de la pila, mientras que el eje y muestra la profundidad de la pila e incluye el seguimiento de la pila. Un único rectángulo en un gráfico tipo llama representa un marco de pila, y el ancho de un marco muestra la frecuencia con la que aparece en las pilas. Para obtener más información acerca de los gráficos tipo llama y cómo usarlos, consulte [Gráficos tipo llama](#).

En Flink, se puede acceder al gráfico tipo llama de un operador a través de la interfaz de usuario web seleccionando un operador y, a continuación, eligiendo la pestaña `FlameGraph`. Una vez que se hayan recolectado suficientes muestras, se mostrará el gráfico tipo llama. A continuación se muestra el `FlameGraph` de la `ProcessFunction`, que tardó mucho tiempo en comprobarse.



Se trata de un gráfico tipo llama muy simple y muestra que todo el tiempo de la CPU se emplea en un vistazo de foreach dentro del `processElement` del operador `ExpensiveFunction`. También se obtiene el número de línea para ayudar a determinar en qué parte de la ejecución código se está realizando.

Se está agotando el tiempo para llegar al punto de control

Si la aplicación no está optimizada o aprovisionada correctamente, los puntos de control pueden fallar. En esta sección se describen los síntomas y los pasos para solucionar esta condición.

Síntomas

Si los puntos de control de su aplicación fallan, `numberOfFailedCheckpoints` será mayor que cero.

Los puntos de control pueden fallar debido a errores directos, como errores de la aplicación, o debido a errores transitorios, como la falta de recursos de la aplicación. Compruebe los registros y las métricas de su aplicación para detectar los siguientes síntomas:

- Errores en el código.
- Errores al acceder a los servicios de dependientes de la aplicación.
- Errores al serializar los datos. Si el serializador predeterminado no puede serializar los datos de la aplicación, la aplicación fallará. Para obtener información sobre el uso de un serializador personalizado en su aplicación, consulte [Serializadores personalizados](#) en la documentación de [Apache Flink](#).

- Errores de falta de memoria.
- Aumentos repentinos o aumentos constantes en las siguientes métricas:
 - `heapMemoryUtilization`
 - `oldGenerationGCTime`
 - `oldGenerationGCCount`
 - `lastCheckpointSize`
 - `lastCheckpointDuration`

Para obtener más información sobre la supervisión de los puntos de control, consulte [Supervisión de los puntos de control](#) en la [documentación de Apache Flink](#).

Causas y soluciones

Los mensajes de error del registro de aplicaciones muestran la causa de las fallas directas. Los errores transitorios pueden tener las siguientes causas:

- Su aplicación tiene un aprovisionamiento de KPU insuficiente. Para obtener información sobre cómo aumentar el aprovisionamiento de la aplicación, consulte. [Escalado](#)
- El tamaño del estado de su aplicación es demasiado grande. Puede supervisar el tamaño del estado de la aplicación mediante la métrica `lastCheckpointSize`.
- Los datos de estado de la aplicación se distribuyen de forma desigual entre las claves. Si su aplicación utiliza el operador `KeyBy`, asegúrese de que los datos entrantes se dividan equitativamente entre las claves. Si la mayoría de los datos se asignan a una sola clave, se crea un cuello de botella que provoca errores.
- La aplicación está experimentando una resistencia en la memoria o en la recopilación de elementos no utilizados. Supervise los aumentos repentinos o valores que aumentan de forma constante de su aplicación en `heapMemoryUtilization`, `oldGenerationGCTime`, y `oldGenerationGCCount`.

Fallo en el punto de control de la aplicación Apache Beam

Si la aplicación Beam está configurada con [ShutdownSourceSaferIdLems](#) establecido en 0ms, es posible que los puntos de control no se activen porque las tareas están en estado "FINALIZADO". En esta sección se describen los síntomas y la resolución de esta condición.

Síntoma

Vaya a los registros de CloudWatch de la aplicación Managed Service para Apache Flink y compruebe si se ha registrado el siguiente mensaje de registro. El siguiente mensaje de registro indica que el punto de control no se pudo activar porque algunas tareas estaban terminadas.

```
{
  "locationInformation":
  "org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordinator",
  "logger": "org.apache.flink.runtime.checkpoint.CheckpointCoordinator",
  "message": "Failed to trigger checkpoint for job your job ID since some
tasks of job your job ID has been finished, abort the checkpoint Failure reason: Not
all required tasks are currently running.",
  "threadName": "Checkpoint Timer",
  "applicationARN": your application ARN,
  "applicationVersionId": "5",
  "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

También se puede encontrar en el panel de control de Flink, donde algunas tareas han pasado al estado "FINALIZADO" y ya no es posible establecer puntos de control.

Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics	FlameGraph						
ID	Bytes Received	Records Received	Bytes Sent	Records Sent	Attempt	Host	Start Time	Duration	Status	More			
0	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	13m 57s	RUNNING	...			
1	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...			
2	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...			
3	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...			
4	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...			

Causa

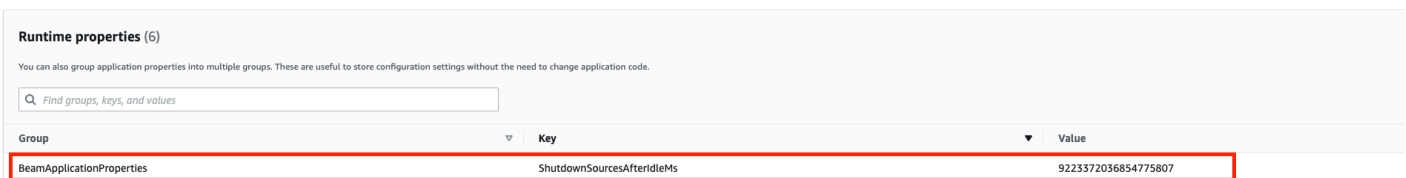
`shutdownSourceAfterIdleMs` es una variable de configuración de Beam que cierra las fuentes que han estado inactivas durante el tiempo configurado de milisegundos. Una vez que se ha cerrado una fuente, establecer puntos de control ya no es posible. Esto podría provocar una [falla en el punto de control](#).

Una de las causas por las que las tareas pasan al estado “FINALIZADO” es cuando `shutdownSourcesAfterIdleMs` está establecido en 0 ms, lo que significa que las tareas que estén inactivas se cerrarán inmediatamente.

Solución

Para evitar que las tareas pasen inmediatamente al estado “FINALIZADO”, defina `shutdownSourceAfterIdleMs` en `Long.MAX_VALUE`. Puede hacer esto de dos formas:

- Opción 1: si su configuración de beam está establecida en la página de configuración de la aplicación Managed Service para Apache Flink, puede añadir un nuevo par clave-valor para configurar `shutdpwnSourcesAfteridleMs` de la siguiente manera:



Runtime properties (6)

You can also group application properties into multiple groups. These are useful to store configuration settings without the need to change application code.

Find groups, keys, and values

Group	Key	Value
BeamApplicationProperties	ShutdownSourcesAfterIdleMs	9223372036854775807

- Opción 2: si su configuración de beam está establecida en el archivo JAR, puede configurar `shutdownSourcesAfterIdleMs` de la siguiente manera:

```
FlinkPipelineOptions options =
PipelineOptionsFactory.create().as(FlinkPipelineOptions.class); // Initialize Beam
Options object

options.setShutdownSourcesAfterIdleMs(Long.MAX_VALUE); // set
shutdownSourcesAfterIdleMs to Long.MAX_VALUE
options.setRunner(FlinkRunner.class);

Pipeline p = Pipeline.create(options); // attach specified
options to Beam pipeline
```

Resistencia

Flink utiliza la resistencia para adaptar la velocidad de procesamiento de los operadores individuales.

El operador puede tener dificultades para seguir procesando el volumen de mensajes que recibe por muchas razones. Es posible que la operación requiera más recursos de CPU de los que dispone el operador. Es posible que el operador espere a que se completen las operaciones de I/O. Si un

operador no puede procesar los eventos con la suficiente rapidez, se genera una resistencia en los operadores de las fases anteriores, lo que repercute en el operador lento. Esto provoca que los operadores de las fases anteriores reduzcan la velocidad, lo que puede propagar aún más la resistencia a la fuente y hacer que la fuente se adapte al rendimiento general de la aplicación, al reducir también la velocidad. Puede encontrar una descripción más detallada de la resistencia y su funcionamiento en [Cómo maneja Apache Flink™ la resistencia](#).

Saber qué operadores de una aplicación son lentos le proporciona información crucial para comprender la causa raíz de los problemas de rendimiento en la aplicación. La información sobre la resistencia se [expone a través del panel de control de Flink](#). Para identificar al operador lento, busque al operador con un valor de resistencia alto que esté más cerca de un receptor (el operador B en el siguiente ejemplo). El operador que provoca la lentitud es entonces uno de los operadores de una fase posterior (en el ejemplo, el operador C). B podría procesar los eventos más rápido, pero enfrenta resistencia, ya que no puede reenviar la salida al operador C que realmente está lento.

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D  
(backpressured 0%)
```

Una vez que haya identificado al operador lento, intente entender por qué es lento. Puede haber una infinidad de razones y, a veces, no es obvio cuál es el problema y su solución puede requerir días de depuración y creación de perfiles. Las siguientes son algunas razones obvias y más comunes, algunas de las cuales se explican con más detalle a continuación:

- El operador realiza operaciones de I/O lentas, por ejemplo, llamadas de red (considere usar `AsyncIO` en su lugar).
- Hay un sesgo en los datos y un operador recibe más eventos que otros (compruébelo observando la cantidad de mensajes que entran y salen de las subtarefas individuales (es decir, instancias del mismo operador) en el panel de control de Flink).
- Se trata de una operación que consume muchos recursos (si no hay un sesgo en los datos, considere la posibilidad de escalar horizontalmente para el trabajo asociado a la CPU o la memoria o aumentarla para el trabajo asociado a la I/O) `ParallelismPerKPU`
- Registro exhaustivo por parte del operador (reduzca el registro al mínimo para la aplicación de producción o, en su lugar, considere enviar los resultados de la depuración a un flujo de datos).

Probando el rendimiento con el receptor de descarte

El [receptor de descarte](#) simplemente ignora todos los eventos que recibe mientras sigue ejecutando la aplicación (una aplicación sin ningún receptor no se ejecuta). Esto resulta muy útil para realizar pruebas de rendimiento, crear perfiles y comprobar si la aplicación se está escalando correctamente. También es un control de estado muy pragmático para comprobar si los receptores están causando la resistencia o la aplicación (pero comprobar las métricas de resistencia suele ser más fácil y sencillo).

Al sustituir todos los receptores de una aplicación por un receptor de descarte y crear una fuente simulada que genere datos que se parezcan a los de producción, se puede medir el rendimiento máximo de la aplicación para una configuración de paralelismo determinada. A continuación, también puede aumentar el paralelismo para comprobar que la aplicación se escala correctamente y que no presenta ningún cuello de botella que solo surja a medida que aumenta el rendimiento (por ejemplo, debido al sesgo de datos).

Sesgo de datos

Una aplicación Flink se ejecuta en un clúster de forma distribuida. Para escalar horizontalmente a varios nodos, Flink utiliza el concepto de flujos con claves, que básicamente significa que los eventos de un flujo se dividen en función de una clave específica, por ejemplo, la identificación del cliente, y Flink puede procesar diferentes particiones en diferentes nodos. Luego, muchos de los operadores de Flink se evalúan en función de estas particiones, por ejemplo, [ventanas con clave](#), [funciones de procesos](#) y [Async I/O](#).

La elección de una clave de partición a menudo depende de la lógica empresarial. Al mismo tiempo, muchas de las prácticas recomendadas, por ejemplo, para [DynamoDB](#) y Spark, también se aplican a Flink, entre las que se incluyen:

- garantizar una alta cardinalidad de las claves de partición
- evitar el sesgo en el volumen de eventos entre las particiones

Puede identificar el sesgo en las particiones comparando los registros recibidos o enviados de las subtareas (es decir, instancias del mismo operador) en el panel de control de Flink. Además, el monitoreo de Managed Service para Apache Flink se puede configurar para mostrar las métricas correspondientes para `numRecordsIn/Out` y `numRecordsInPerSecond/OutPerSecond` a nivel de subtarea.

Sesgo de estado

En el caso de los operadores con estado, es decir, los operadores que mantienen el estado para su lógica empresarial, como ventanas, el sesgo de datos siempre conduce a un sesgo de estado. Algunas subtareas reciben más eventos que otras debido al sesgo de datos y, por lo tanto, también mantienen más datos en estado. Sin embargo, incluso en el caso de una aplicación que tenga particiones equilibradas de manera uniforme, puede haber un sesgo en la cantidad de datos que se conservan en el estado. Por ejemplo, en el caso de las ventanas de sesión, algunos usuarios y sesiones, respectivamente, pueden durar mucho más que otros. Si las sesiones más largas forman parte de la misma partición, se puede producir un desequilibrio en el tamaño del estado que mantienen las distintas subtareas del mismo operador.

El sesgo de estado no solo aumenta los recursos de memoria y disco que requieren las subtareas individuales, sino que también puede disminuir el rendimiento general de la aplicación. Cuando una aplicación toma un punto de control o un punto de guardado, el estado del operador se mantiene en Amazon S3, para proteger el estado contra errores de nodos o clústeres. Durante este proceso (especialmente con la semántica exactamente una vez, que está habilitada de forma predeterminada en Managed Service para Apache Flink), el procesamiento se detiene desde una perspectiva externa hasta que se complete el punto de control o el punto de guardado. Si hay un sesgo de datos, el tiempo necesario para completar la operación puede estar limitado por una sola subtask que haya acumulado una cantidad de estado particularmente alta. En casos extremos, la captura de puntos de control o puntos de almacenamiento puede fallar debido a que una sola subtask no puede mantener su estado.

De forma similar al sesgo de datos, el sesgo de estado puede ralentizar considerablemente una aplicación.

Para identificar el sesgo de estado, puede aprovechar el panel de control de Flink. Busque un punto de control o un punto de guardado reciente y compare en los detalles la cantidad de datos que se han almacenado para cada subtask.

Integración con recursos en diferentes regiones

Puede habilitar usando `StreamingFileSink` para escribir en un bucket de Amazon S3 de una región diferente a la de su aplicación Managed Service para Apache Flink mediante una configuración necesaria para la replicación entre regiones en la configuración de Flink. [Para ello, presente un ticket de soporte en el AWS Support Centro.](#)

Historial de documentos de Amazon Managed Service para Apache Flink

En la siguiente tabla se describen los cambios importantes que se han realizado en la documentación desde la última versión de Managed Service para Apache Flink.

- Versión de la API: 2018-05-23
- Última actualización de la documentación: 30 de agosto de 2023

Cambio	Descripción	Fecha
Kinesis Data Analytics ahora se conoce como Managed Service para Apache Flink	No hay cambios en los puntos finales del servicio, las API, la interfaz de línea de comandos, las políticas de acceso de IAM, las CloudWatch métricas o los AWS paneles de facturación. Las aplicaciones existentes continuarán funcionando como antes. Para obtener más información, consulte ¿Qué es Managed Service para Apache Flink?	30 de agosto de 2023
Compatibilidad con Apache Flink, versión 1.15.2	Managed Service para Apache Flink ahora es compatible con las aplicaciones que utilizan la versión 1.15.2 de Apache Flink. Cree aplicaciones de Kinesis Data Analytics usando la API de tablas de Apache Flink. Para obtener más información, consulte Creación de aplicaciones .	22 de noviembre de 2022

Cambio	Descripción	Fecha
Compatibilidad con Apache Flink, versión 1.13.2	Managed Service para Apache Flink ahora es compatible con las aplicaciones que utilizan la versión 1.13.2 de Apache Flink. Cree aplicaciones de Kinesis Data Analytics usando la API de tablas de Apache Flink. Para obtener más información, consulte Introducción: Flink 1.13.2 .	13 de octubre de 2021
Compatibilidad con Python	Managed Service para Apache Flink ahora es compatible con aplicaciones que utilizan Python con la API de tablas de Apache Flink y SQL. Para obtener más información, consulte Uso de Python .	25 de marzo de 2021
Compatibilidad con Apache Flink 1.11.1	Managed Service para Apache Flink ahora es compatible con las aplicaciones que utilizan Apache Flink 1.11.1. Cree aplicaciones de Kinesis Data Analytics usando la API de tablas de Apache Flink. Para obtener más información, consulte Creación de aplicaciones .	19 de noviembre de 2020

Cambio	Descripción	Fecha
Panel de Apache Flink	Utilice el panel de Apache Flink para supervisar el estado y el rendimiento de las aplicaciones. Para obtener más información, consulte Panel de Apache Flink .	19 de noviembre de 2020
Consumidor de EFO	Cree aplicaciones que utilicen un consumidor de distribución mejorada (EFO) para leer desde un flujo de datos de Kinesis. Para obtener más información, consulte Consumidor de EFO .	6 de octubre de 2020
Apache Beam	Cree aplicaciones que utilicen Apache Beam para procesar datos de streaming. Para obtener más información, consulte Uso de CloudFormation con Managed Service para Apache Flink .	15 de septiembre de 2020
Rendimiento	Cómo solucionar los problemas de rendimiento de las aplicaciones y cómo crear una aplicación que funcione. Para obtener más información, consulte Desempeño .	21 de julio de 2020

Cambio	Descripción	Fecha
Almacén de claves personalizado	Cómo acceder a un clúster de Amazon MSK que utiliza un almacén de claves personalizado para el cifrado en tránsito. Para obtener más información, consulte Truststore personalizado .	10 de junio de 2020
CloudWatch Alarmas	Recomendaciones para crear CloudWatch alarmas con Managed Service for Apache Flink. Para obtener más información, consulte Alarmas .	5 de junio de 2020
Nuevas métricas CloudWatch	El servicio gestionado para Apache Flink ahora emite 22 métricas a Amazon CloudWatch Metrics. Para obtener más información, consulte Métricas y dimensiones en Managed Service para Apache Flink .	12 de mayo de 2020
Métricas personalizadas CloudWatch	Defina métricas específicas de la aplicación y emítalas a Amazon CloudWatch Metrics. Para obtener más información, consulte Métricas personalizadas .	12 de mayo de 2020

Cambio	Descripción	Fecha
Ejemplo: leer desde un flujo de Kinesis en una cuenta diferente	Conozca cómo acceder a un flujo de Kinesis en una cuenta de AWS diferente en su aplicación Managed Service para Apache Flink. Para obtener más información, consulte entre cuentas .	30 de marzo de 2020
Compatibilidad con Apache Flink 1.8.2	Managed Service para Apache Flink ahora es compatible con las aplicaciones que utilizan Apache Flink 1.8.2. Utilice el Streaming FileSink conector Flink para escribir la salida directamente en S3. Para obtener más información, consulte Creación de aplicaciones .	17 de diciembre de 2019
VPC de Managed Service para Apache Flink	Configure una aplicación de Managed Service para Apache Flink para que se conecte a una nube privada virtual. Para obtener más información, consulte Uso de una Amazon VPC .	25 de noviembre de 2019
Mejores prácticas de Managed Service para Apache Flink	Mejores prácticas para crear y administrar aplicaciones de Managed Service para Apache Flink. Para obtener más información, consulte Prácticas recomendadas .	14 de octubre de 2019

Cambio	Descripción	Fecha
Analizar registros de aplicaciones de Managed Service para Apache Flink	Utilice CloudWatch Logs Insights para supervisar su servicio gestionado para la aplicación Apache Flink. Para obtener más información, consulte Análisis de registros .	26 de junio de 2019
Propiedades de tiempo de ejecución de aplicaciones Managed Service para Apache Flink	Trabaje con propiedades de tiempo de ejecución en Managed Service para Apache Flink. Para obtener más información, consulte Propiedades de tiempo de ejecución .	24 de junio de 2019
Etiquetado de aplicaciones Managed Service para Apache Flink	Utilice las etiquetas de las aplicaciones para determinar los costos de cada aplicación, para controlar el acceso o para los fines que defina el usuario. Para obtener más información, consulte Uso del etiquetado .	8 de mayo de 2019
Aplicaciones de ejemplo de Managed Service para Apache Flink	Ejemplos de aplicaciones de Managed Service para Apache Flink que muestran los operadores de ventanas y la escritura de los resultados en Logs. CloudWatch Para obtener más información, consulte Ejemplos .	1 de mayo de 2019

Cambio	Descripción	Fecha
Registro de llamadas de API de Managed Service para Apache Flink con AWS CloudTrail	Managed Service para Apache Flink está integrado con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, rol o servicio de AWS en Managed Service para Apache Flink. Para obtener más información, consulte Uso de AWS CloudTrail .	22 de marzo de 2019
Creación de una aplicación (receptor de Kinesis Data Firehose)	Intente crear una aplicación de Managed Service para Apache Flink con un Amazon Kinesis Data Streams como origen y un flujo de Amazon Kinesis Data Firehose como receptor. Para obtener más información, consulte Receptor de Kinesis Data Firehose .	13 de diciembre de 2018
Versión pública	Esta es la versión inicial de la Guía para desarrolladores de Managed Service para Apache Flink para aplicaciones Java.	27 de noviembre de 2018

Ejemplo de código de API de Managed Service para Apache Flink

Este tema contiene ejemplos de bloques de solicitud de acciones de Managed Service para Apache Flink.

Para usar JSON como entrada para una acción con la AWS Command Line Interface (AWS CLI), guarde la solicitud en un archivo JSON. A continuación, pase el nombre del archivo a la acción mediante el parámetro `--cli-input-json`.

En el siguiente ejemplo se muestra cómo utilizar un archivo JSON con una acción.

```
$ aws kinesisanalyticstv2 start-application --cli-input-json file://start.json
```

Para obtener más información sobre cómo usar JSON con la AWS CLI, consulte [Generar el esqueleto de la CLI y los parámetros JSON de entrada de CLI](#) en la Guía del usuario de AWS Command Line Interface.

Temas

- [AddApplicationCloudWatchLoggingOption](#)
- [AddApplicationInput](#)
- [AddApplicationInputProcessingConfiguration](#)
- [AddApplicationOutput](#)
- [AddApplicationReferenceDataSource](#)
- [AddApplicationVpcConfiguration](#)
- [CreateApplication](#)
- [CreateApplicationSnapshot](#)
- [DeleteApplication](#)
- [DeleteApplicationCloudWatchLoggingOption](#)
- [DeleteApplicationInputProcessingConfiguration](#)
- [DeleteApplicationOutput](#)
- [DeleteApplicationReferenceDataSource](#)
- [DeleteApplicationSnapshot](#)

- [DeleteApplicationVpcConfiguration](#)
- [DescribeApplication](#)
- [DescribeApplicationSnapshot](#)
- [DiscoverInputSchema](#)
- [ListApplications](#)
- [ListApplicationSnapshots](#)
- [StartApplication](#)
- [StopApplication](#)
- [UpdateApplication](#)

AddApplicationCloudWatchLoggingOption

El siguiente código de solicitud de ejemplo para la acción [AddApplicationCloudWatchLoggingOption](#) añade una opción de registro de Amazon CloudWatch a una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CloudWatchLoggingOption": {
    "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-
group:log-stream:My-LogStream"
  },
  "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

El siguiente código de solicitud de ejemplo para la acción [AddApplicationInput](#) añade una entrada de aplicación a una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "Input": {
    "InputParallelism": {
      "Count": 2
    },
  },
}
```

```

    "InputSchema": {
      "RecordColumns": [
        {
          "Mapping": "$.TICKER",
          "Name": "TICKER_SYMBOL",
          "SqlType": "VARCHAR(50)"
        },
        {
          "SqlType": "REAL",
          "Name": "PRICE",
          "Mapping": "$.PRICE"
        }
      ],
      "RecordEncoding": "UTF-8",
      "RecordFormat": {
        "MappingParameters": {
          "JSONMappingParameters": {
            "RecordRowPath": "$"
          }
        },
        "RecordFormatType": "JSON"
      }
    },
    "KinesisStreamsInput": {
      "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
    }
  }
}

```

AddApplicationInputProcessingConfiguration

El siguiente código de solicitud de ejemplo para la acción

[AddApplicationInputProcessingConfiguration](#) añade una configuración de procesamiento de entradas de aplicación a una aplicación de Managed Service para Apache Flink:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "InputId": "2.1",
  "InputProcessingConfiguration": {
    "InputLambdaProcessor": {

```

```
    "ResourceARN": "arn:aws:lambda:us-  
east-1:012345678901:function:MyLambdaFunction"  
  }  
}  
}
```

AddApplicationOutput

El siguiente código de solicitud de ejemplo para la acción [AddApplicationOutput](#) añade un flujo de datos de Kinesis como salida de la aplicación a una aplicación de Managed Service para Apache Flink:

```
{  
  "ApplicationName": "MyApplication",  
  "CurrentApplicationVersionId": 2,  
  "Output": {  
    "DestinationSchema": {  
      "RecordFormatType": "JSON"  
    },  
    "KinesisStreamsOutput": {  
      "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/  
ExampleOutputStream"  
    },  
    "Name": "DESTINATION_SQL_STREAM"  
  }  
}
```

AddApplicationReferenceDataSource

El siguiente código de solicitud de ejemplo para la acción [AddApplicationReferenceDataSource](#) añade el origen de datos de referencia de la aplicación CSV a una aplicación de Managed Service para Apache Flink:

```
{  
  "ApplicationName": "MyApplication",  
  "CurrentApplicationVersionId": 5,  
  "ReferenceDataSource": {  
    "ReferenceSchema": {  
      "RecordColumns": [  
        {
```

```

        "Mapping": "$.TICKER",
        "Name": "TICKER",
        "SqlType": "VARCHAR(4)"
    },
    {
        "Mapping": "$.COMPANYNAME",
        "Name": "COMPANY_NAME",
        "SqlType": "VARCHAR(40)"
    },
],
"RecordEncoding": "UTF-8",
"RecordFormat": {
    "MappingParameters": {
        "CSVMappingParameters": {
            "RecordColumnDelimiter": " ",
            "RecordRowDelimiter": "\r\n"
        }
    },
    "RecordFormatType": "CSV"
}
},
"S3ReferenceDataSource": {
    "BucketARN": "arn:aws:s3:::MyS3Bucket",
    "FileKey": "TickerReference.csv"
},
"TableName": "string"
}
}

```

AddApplicationVpcConfiguration

El siguiente código de solicitud de ejemplo para la acción [AddApplicationVpcConfiguration](#) añade una configuración de VPC a una aplicación existente:

```

{
    "ApplicationName": "MyApplication",
    "CurrentApplicationVersionId": 9,
    "VpcConfiguration": {
        "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
        "SubnetIds": [ "subnet-0123456789abcdef0" ]
    }
}

```

CreateApplication

El siguiente código de solicitud de ejemplo para la acción [CreateApplication](#) crea una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My-Application-Description",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "CloudWatchLoggingOptions": [
    {
      "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-stream:My-LogStream"
    }
  ],
  "ApplicationConfiguration": {
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-east-1",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-east-1"
          }
        }
      ]
    }
  },
  "ApplicationCodeConfiguration": {
    "CodeContent": {
      "S3ContentLocation": {
        "BucketARN": "arn:aws:s3:::mybucket",
        "FileKey": "myflink.jar",
        "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    }
  },
  "CodeContentType": "ZIPFILE"
}
```

```
    },
    "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    }
  }
}
```

CreateApplicationSnapshot

El siguiente código de solicitud de ejemplo para la acción [CreateApplicationSnapshot](#) crea una instantánea del estado de la aplicación:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotName": "MySnapshot"
}
```

DeleteApplication

El siguiente código de solicitud de ejemplo para la acción [DeleteApplication](#) elimina una aplicación de Managed Service para Apache Flink:

```
{"ApplicationName": "MyApplication",
 "CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

El siguiente código de solicitud de ejemplo para la acción [DeleteApplicationCloudWatchLoggingOption](#) elimina una opción de registro de Amazon CloudWatch de una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
```

```
"CloudWatchLoggingOptionId": "3.1"
"CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

El siguiente código de solicitud de ejemplo para la acción

[DeleteApplicationInputProcessingConfiguration](#) elimina una configuración de procesamiento de entradas de una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "InputId": "2.1"
}
```

DeleteApplicationOutput

El siguiente código de solicitud de ejemplo para la acción [DeleteApplicationOutput](#) elimina una salida de la aplicación de una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

El siguiente código de solicitud de ejemplo para la acción [DeleteApplicationReferenceDataSource](#) elimina el origen de datos de referencia de la aplicación de una aplicación de Managed Service para Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

El siguiente código de solicitud de ejemplo para la acción [DeleteApplicationSnapshot](#) elimina una instantánea del estado de la aplicación:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotCreationTimestamp": 12345678912,
  "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

El siguiente código de solicitud de ejemplo para la acción [DeleteApplicationVpcConfiguration](#) elimina una configuración de VPC existente de una aplicación:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfigurationId": "1.1"
}
```

DescribeApplication

El siguiente código de solicitud de ejemplo para la acción [DescribeApplication](#) proporciona detalles sobre una aplicación de Managed Service para Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

El siguiente código de solicitud de ejemplo para la acción [DescribeApplicationSnapshot](#) proporciona detalles sobre una instantánea del estado de la aplicación:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotName": "MySnapshot"
}
```



```
}
```

DiscoverInputSchema

El siguiente código de solicitud de ejemplo para la acción [DiscoverInputSchema](#) genera un esquema a partir de un origen de streaming:

```
{
  "InputProcessingConfiguration": {
    "InputLambdaProcessor": {
      "ResourceARN": "arn:aws:lambda:us-east-1:012345678901:function:MyLambdaFunction"
    }
  },
  "InputStartingPositionConfiguration": {
    "InputStartingPosition": "NOW"
  },
  "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
  "S3Configuration": {
    "BucketARN": "string",
    "FileKey": "string"
  },
  "ServiceExecutionRole": "string"
}
```

El siguiente código de solicitud de ejemplo para la acción [DiscoverInputSchema](#) genera un esquema a partir de una fuente de referencia:

```
{
  "S3Configuration": {
    "BucketARN": "arn:aws:s3:::mybucket",
    "FileKey": "TickerReference.csv"
  },
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

El siguiente código de solicitud de ejemplo para la acción [ListApplications](#) proporciona una lista de las aplicaciones de Managed Service para Apache Flink de su cuenta:

```
{
  "ExclusiveStartApplicationName": "MyApplication",
  "Limit": 50
}
```

ListApplicationSnapshots

El siguiente código de solicitud de ejemplo para la acción [ListApplicationSnapshots](#) proporciona una lista de instantáneas del estado de la aplicación:

```
{"ApplicationName": "MyApplication",
  "Limit": 50,
  "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

El siguiente código de solicitud de ejemplo para la acción [StartApplication](#) inicia una aplicación de Managed Service para Apache Flink y carga el estado de la aplicación a partir de la última instantánea (de haberla):

```
{
  "ApplicationName": "MyApplication",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}
```

StopApplication

El siguiente código de solicitud de ejemplo para la acción [API_StopApplication](#) detiene una aplicación de Managed Service para Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

El siguiente código de solicitud de ejemplo para la acción [UpdateApplication](#) actualiza una aplicación de Managed Service para Apache Flink para cambiar la ubicación del código de la aplicación:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentTypeUpdate": "ZIPFILE",
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
          "FileKeyUpdate": "my_new_code.zip",
          "ObjectVersionUpdate": "2"
        }
      }
    }
  }
}
```

Referencia de API de Managed Service para Apache Flink

Para obtener información sobre las API que proporciona Managed Service para Apache Flink, consulte la [Referencia de API de Managed Service para Apache Flink](#).