aws

Guia do desenvolvedor do Managed Service for Apache Flink

Managed Service for Apache Flink

Managed Service for Apache Flink: Guia do desenvolvedor do Managed Service for Apache Flink

Table of Contents

Anteriormente, o Amazon Managed Service for Apache Flink era conhecido como Amazon Kinesis Data Analytics for Apache Flink.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.

O que é Amazon Managed Service for Apache Flink?

Com o Amazon Managed Service para Apache Flink, você pode usar Java, Scala, Python ou SQL para processar e analisar dados de streaming. O serviço permite que você crie e execute código em fontes de streaming e fontes estáticas para realizar análises de séries temporais, alimentar painéis e métricas em tempo real.

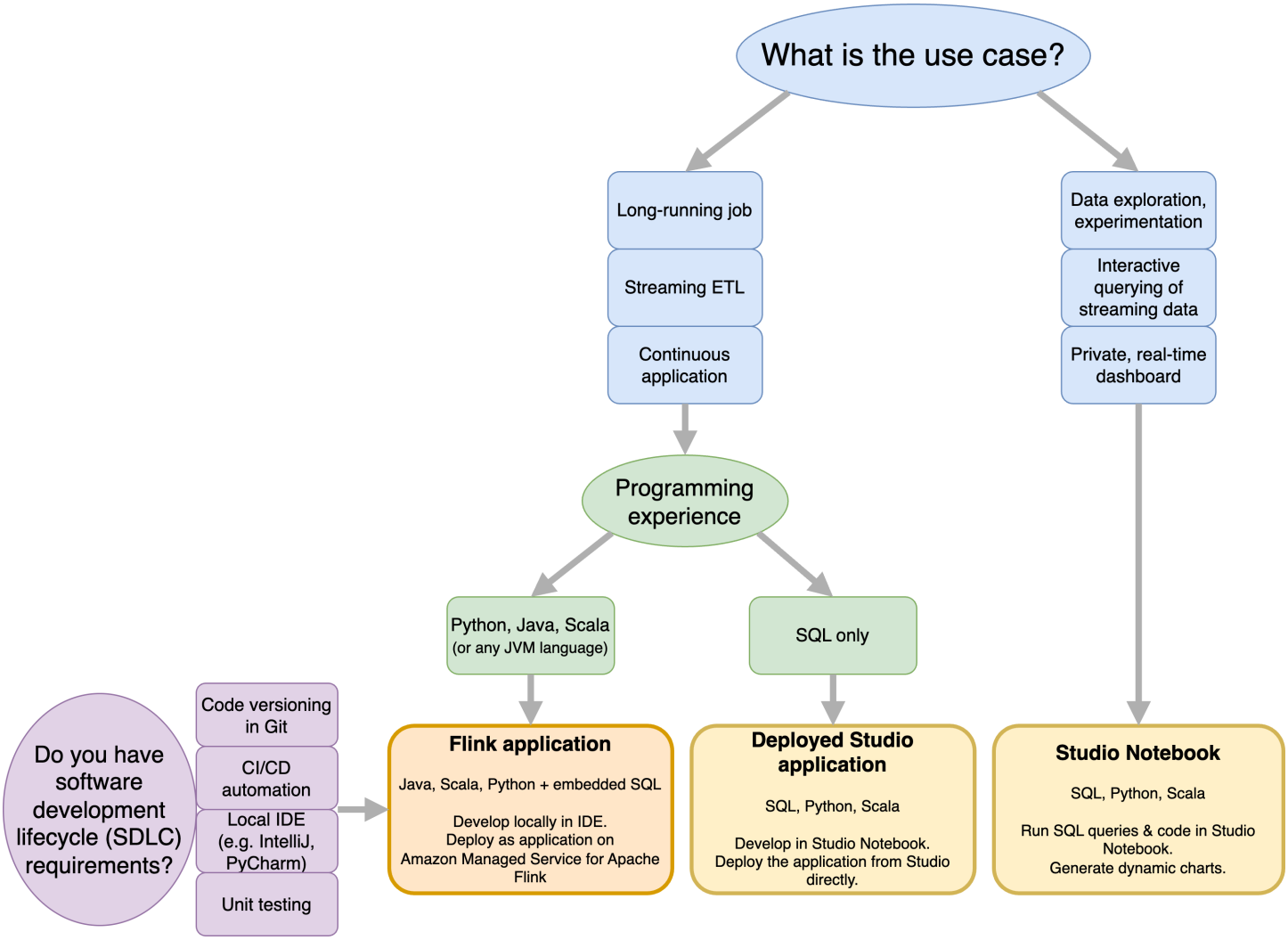
Você pode criar aplicativos com a linguagem de sua escolha no Managed Service for Apache Flink usando bibliotecas de código aberto baseadas no Apache Flink. O Apache Flink é uma estrutura popular e um mecanismo para o processamento de fluxos de dados.

O Managed Service for Apache Flink fornece a infraestrutura subjacente para seus aplicativos Apache Flink. Ele lida com os principais recursos, como provisionamento de recursos computacionais, resiliência de failover AZ, computação paralela, escalabilidade automática e backups de aplicativos (implementados como pontos de verificação e instantâneos). Você pode usar os recursos de programação de alto nível do Flink (como operadores, funções, fontes e coletores) da mesma forma que os usa ao hospedar você mesmo a infraestrutura do Flink.

Escolhendo o serviço gerenciado para o Apache Flink ou o serviço gerenciado para o Apache Flink Studio

Você tem duas opções para executar suas tarefas do Flink com o Amazon Managed Service para Apache Flink. Com o Managed Service for Apache Flink, você cria aplicativos Flink em Java, Scala ou Python (e SQL incorporado) usando um IDE de sua escolha e as APIs Apache Flink Datastream ou Table. Com o Managed Service for Apache Flink Studio, você pode consultar interativamente fluxos de dados em tempo real e criar e executar facilmente aplicativos de processamento de streams usando SQL, Python e Scala padrão.

Você pode selecionar o método mais adequado ao seu caso de uso. Se você não tiver certeza, esta seção oferecerá orientação de alto nível para ajudá-lo.



Antes de decidir se quer usar o Amazon Managed Service para Apache Flink ou o Amazon Managed Service para Apache Flink Studio, você deve considerar seu caso de uso.

Se você planeja operar um aplicativo de longa execução que executará cargas de trabalho como Streaming ETL ou Continuous Applications, considere usar o Managed Service for Apache Flink. Isso ocorre porque você pode criar seu aplicativo Flink usando as APIs do Flink diretamente no IDE de sua escolha. Desenvolver localmente com seu IDE também garante que você possa aproveitar processos e ferramentas comuns do ciclo de vida de desenvolvimento de software (SDLC), como controle de versão de código no Git, automação de CI/CD ou teste unitário.

Se você estiver interessado na exploração de dados ad-hoc, quiser consultar dados de streaming de forma interativa ou criar painéis privados em tempo real, o Managed Service for Apache Flink Studio ajudará você a atingir essas metas com apenas alguns cliques. Usuários familiarizados com o SQL podem considerar a implantação de um aplicativo de longa execução diretamente do Studio.

> **Note**
>
> Você pode promover seu notebook Studio em um aplicativo de longa duração. No entanto, se você quiser se integrar às suas ferramentas do SDLC, como controle de versão de código no Git e automação de CI/CD, ou técnicas como teste de unidade, recomendamos o Managed Service for Apache Flink usando o IDE de sua escolha.

Escolhendo quais APIs do Apache Flink usar no Managed Service for Apache Flink

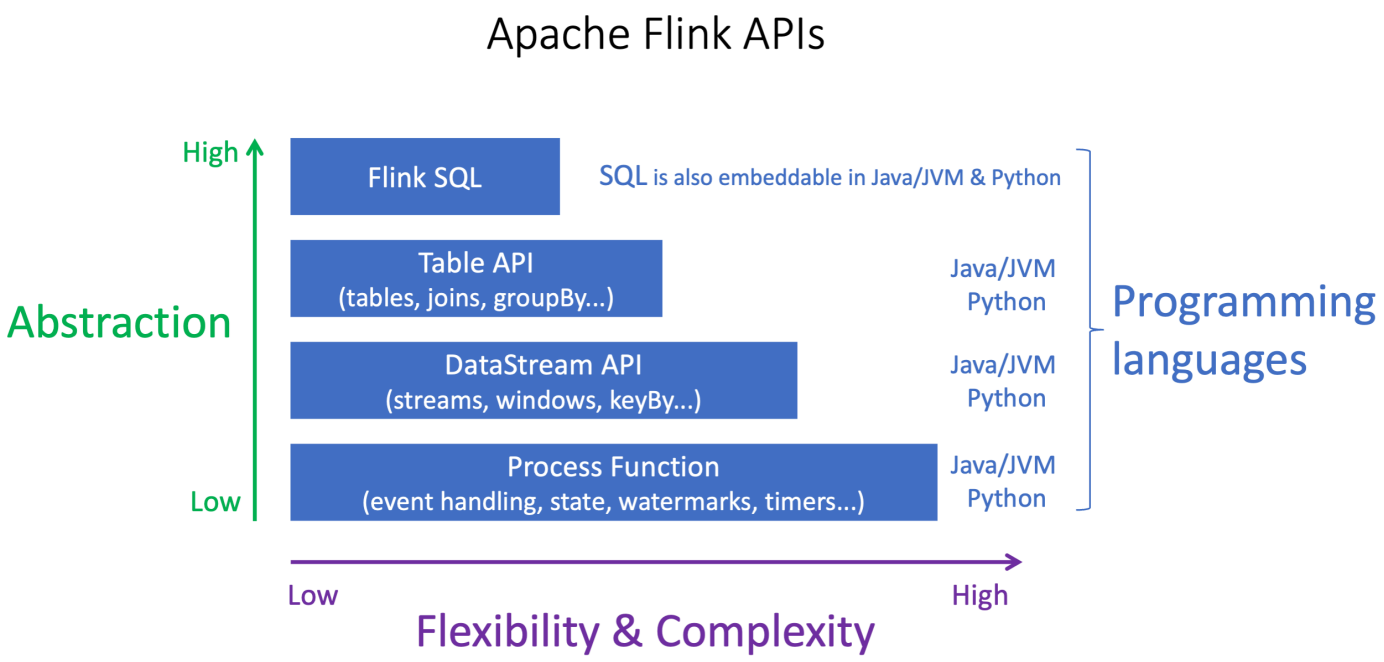
Você pode criar aplicativos usando Java, Python e Scala no Managed Service for Apache Flink usando as APIs do Apache Flink em um IDE de sua escolha. Você pode encontrar orientações sobre como criar aplicativos usando a API Flink Datastream and Table na documentação. Você pode selecionar o idioma no qual você cria seu aplicativo Flink e as APIs que você usa para melhor atender às necessidades de seu aplicativo e operações. Se você não tiver certeza, esta seção fornece orientação de alto nível para ajudá-lo.

Escolhendo uma API Flink

As APIs do Apache Flink têm diferentes níveis de abstração que podem afetar a forma como você decide criar seu aplicativo. Eles são expressivos e flexíveis e podem ser usados juntos para criar seu aplicativo. Você não precisa usar apenas uma API do Flink. Você pode aprender mais sobre as APIs do Flink na documentação do Apache Flink.

O Flink oferece quatro níveis de abstração de API: Flink SQL, API de tabela, DataStream API e função de processo, que são usados em conjunto com a API. DataStream Tudo isso é compatível com o Amazon Managed Service para Apache Flink. É recomendável começar com um nível mais alto de abstração sempre que possível. No entanto, alguns recursos do Flink só estão disponíveis com a API Datastream, na qual você pode criar seu aplicativo em Java, Python ou Scala. Você deve considerar o uso da API Datastream se:

- Você precisa de um controle refinado sobre o estado
- Você deseja aproveitar a capacidade de chamar um banco de dados externo ou endpoint de forma assíncrona (por exemplo, para inferência)
- Você deseja usar temporizadores personalizados



> **Note**
>
> Escolha de um idioma com a API Datastream:
>
> - O SQL pode ser incorporado em qualquer aplicativo Flink, independentemente da linguagem de programação escolhida.
> - Se você planeja usar a DataStream API, nem todos os conectores são compatíveis com Python.
> - Se você precisar de baixa latência/alto rendimento, considere o Java/Scala, independentemente da API.
> - Se você planeja usar o Async IO na API Process Functions, precisará usar Java.

Conceitos básicos

Você pode começar criando um aplicativo do Managed Service for Apache Flink que lê e processa continuamente dados em transmissão. Em seguida, crie seu código usando o IDE de sua escolha e teste-o com dados de transmissão ao vivo. Você também pode configurar destinos para os quais o Managed Service for Apache Flink enviará os resultados.

Para começar, recomendamos que você leia as seguintes seções:

- Como funciona o Managed Service for Apache Flink
- Introdução ao Amazon Managed Service para Apache Flink (DataStream API)

Como alternativa, você pode começar criando um serviço gerenciado para o notebook Apache Flink Studio que permite consultar interativamente fluxos de dados em tempo real e criar e executar facilmente aplicativos de processamento de fluxo usando SQL, Python e Scala padrão. Com alguns cliques no AWS Management Console, você pode iniciar um caderno de notas sem servidor para consultar fluxos de dados e obter resultados em segundos. Para começar, recomendamos que você leia as seguintes seções:

- Usar um notebook Studio com Managed Service for Apache Flink
- Criar um notebook Studio

Como funciona o Managed Service for Apache Flink

O Managed Service for Apache Flink é um serviço Amazon totalmente gerenciado que permite usar um aplicativo Apache Flink para processar dados de transmissão.

Programar seu aplicativo Apache Flink

Um aplicativo Apache Flink é um aplicativo Java ou Scala criado com a estrutura Apache Flink. Você cria e constrói seu aplicativo Apache Flink localmente.

Os aplicativos usam principalmente a API DataStream ou a API de tabela. As outras APIs do Apache Flink também estão disponíveis para uso, mas são menos usadas na criação de aplicativos de transmissão.

Os atributos das duas APIs são:

API de fluxo de dados

O modelo de programação da API DataStream do Apache Flink é baseado em dois componentes:

- Fluxo de dados: a representação estruturada de um fluxo contínuo de registros de dados.
- Operador de transformação: usa um ou mais fluxos de dados como entrada e produz um ou mais fluxos de dados como saída.

Os aplicativos criados com a API DataStream fazem o seguinte:

- Leem dados de uma fonte de dados (como um fluxo do Kinesis ou um tópico do Amazon MSK).
- Aplicam transformações aos dados, como filtragem, agregação ou enriquecimento.
- Gravam os dados transformados em um coletor de dados.

Os aplicativos que usam a API DataStream podem ser escritos em Java ou Scala e podem ser lidos de um fluxo de dados Kinesis, de um tópico do Amazon MSK ou de uma fonte personalizada.

Seu aplicativo processa dados usando um conector. O Apache Flink usa os tipos de conectores a seguir:

- Fonte: um conector usado para ler dados externos.
- Coletor: um conector usado para gravar em locais externos.

- Operador: um conector usado para processar dados dentro do aplicativo.

Um aplicativo típico consiste em pelo menos um fluxo de dados com uma fonte, um fluxo de dados com um ou mais operadores e pelo menos um coletor de dados.

Para obter mais informações sobre como usar a API DataStream, consulte API de fluxo de dados.

API de tabela

O modelo de programação da API de tabela do Apache Flink é baseado nos seguintes componentes:

- Ambiente de tabela: uma interface para dados subjacentes usado para criar e hospedar uma ou mais tabelas.
- Tabela: um objeto que fornece acesso a uma tabela ou exibição SQL.
- Fonte da tabela: usada para ler dados de uma fonte externa, como um tópico do Amazon MSK.
- Função de tabela: uma consulta SQL ou chamada de API usada para transformar dados.
- Coletor de tabela: usado para gravar dados em um local externo, como um bucket do Amazon S3.

Os aplicativos criados com a API de tabela fazem o seguinte:

- Criam um `TableEnvironment` conectando-se a um `Table Source`.
- Criam uma tabela no `TableEnvironment` usando as funções de consultas SQL ou API de tabela.
- Executam uma consulta na tabela usando a API de tabela ou SQL
- Aplicam transformações nos resultados da consulta usando funções de API de tabela ou consultas SQL.
- Gravam os resultados da consulta ou função em um `Table Sink`.

Os aplicativos que usam a API de tabela podem ser escritos em Java ou Scala e podem consultar dados usando chamadas de API ou consultas SQL.

Para obter mais informações sobre como usar a API de tabela, consulte API de tabela.

Criar seu aplicativo Managed Service for Apache Flink

O Managed Service for Apache Flink é um serviço AWS que cria um ambiente para hospedar seu aplicativo Apache Flink e fornece as seguintes configurações:

- Propriedades de runtime: parâmetros que você pode fornecer ao seu aplicativo. Você pode alterar esses parâmetros sem recompilar o código do aplicativo.
- Tolerância a falhas: como seu aplicativo se recupera de interrupções e reinicializações.
- Registro e Monitoramento: como seu aplicativo registra eventos no CloudWatch Logs.
- Escalabilidade: como seu aplicativo provisiona recursos de computação.

Você pode criar seu aplicativo Managed Service for Apache Flink usando o console ou o AWS CLI. Para começar a criar um aplicativo Managed Service for Apache Flink, consulte Introdução (DataStream API).

Criar um aplicativo Managed Service for Apache Flink

Este tópico contém informações sobre como criar um Managed Service for Apache Flink.

Este tópico contém as seguintes seções:

- Criar seu código de aplicativo Managed Service for Apache Flink
- Criar seu aplicativo Managed Service for Apache Flink
- Iniciar seu aplicativo Managed Service for Apache Flink
- Verificar seu aplicativo Managed Service for Apache Flink

Criar seu código de aplicativo Managed Service for Apache Flink

Esta seção descreve os componentes que você usa para criar o código do aplicativo Managed Service for Apache Flink.

Recomendamos que você use a versão mais recente suportada do Apache Flink para o seu código do aplicativo. A versão mais recente do Apache Flink suportada pelo Managed Service for Apache Flink é 1.15.2. Para obter informações sobre a atualização de aplicativos Managed Service for Apache Flink, consulte Atualização de aplicativos.

Você cria o código do seu aplicativo usando o Apache Maven. Um projeto Apache Maven usa um arquivo `pom.xml` para especificar as versões dos componentes que ele usa.

> **Note**
>
> O Managed Service for Apache Flink suporta arquivos JAR de até 512 MB de tamanho. Se você usar um arquivo JAR maior do que isso, seu aplicativo falhará ao iniciar.

Use as seguintes versões de componentes para os aplicativos do Managed Service for Apache Flink:

Componente	Version (Versão)
Java	11 (recomendada)
Scala	Veja a nota de dissociação do Scala abaixo
Serviço gerenciado para Apache Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
AWSConector Kinesis () flink-connector-kinesis	1.15.2
Apache Beam (somente aplicativos Beam)	2.33.0, com Jackson versão 2.12.2

A partir da versão 1.15, o Flink é compatível com qualquer versão do Scala. Agora, os aplicativos podem usar a API Java de qualquer versão do Scala. Você precisará empacotar a biblioteca padrão Scala de sua escolha em seus aplicativos Scala.

Para obter um exemplo de arquivo `pom.xml` para um aplicativo do Managed Service for Apache Flink que usa o Apache Flink versão 1.15.2, consulte Conceitos básicos do aplicativo Managed Service for Apache Flink.

Para obter informações sobre como criar um aplicativo Managed Service for Apache Flink que usa Apache Beam, consulte Usar o Apache Beam.

Especificar a versão do Apache Flink do seu aplicativo

Ao usar o runtime do Managed Service for Apache Flink versão 1.1.0 e posterior, você especifica a versão do Apache Flink que seu aplicativo usa ao compilar seu aplicativo. Você fornece a versão do Apache Flink com o parâmetro `-Dflink.version` da seguinte forma:

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

Para criar aplicativos com versões mais antigas do Apache Flink, consulte Versões anteriores.

Criar seu aplicativo Managed Service for Apache Flink

Depois de criar o código do aplicativo, faça o seguinte para criar seu aplicativo Managed Service for Apache Flink:

- Faça upload do código do aplicativo: faça upload do código do aplicativo em um bucket do Amazon S3. Ao criar o aplicativo, você especifica o nome do bucket do S3 e o nome do objeto do código do aplicativo. Para ver um tutorial que mostra como fazer upload do código do seu aplicativo, consulte the section called “faça o upload do código Java Apache Flink Streaming” no tutorial Introdução (DataStream API).
- Crie seu aplicativo Managed Service for Apache Flink: use um dos métodos a seguir para criar seu aplicativo Managed Service for Apache Flink:
  - Crie seu aplicativo Managed Service for Apache Flink usando o console AWS: é possível criar e configurar seu aplicativo usando o console AWS.

    Quando você cria seu aplicativo usando o console, os recursos dependentes do seu aplicativo (como fluxos de CloudWatch registros, funções do IAM e políticas do IAM) são criados para você.

    Ao criar seu aplicativo usando o console, você especifica qual versão do Apache Flink seu aplicativo usa selecionando-a no menu suspenso na página Managed Service for Apache Flink - Criar aplicativo.

    Para obter um tutorial sobre como usar o console para criar um aplicativo, consulte the section called “Criar e executar o aplicativo (console)” no Introdução (DataStream API) tutorial.
  - Crie seu aplicativo Managed Service for Apache Flink usando o CLI AWS: é possível criar e configurar seu aplicativo usando o CLI AWS.

    Ao criar seu aplicativo usando a CLI, você também deve criar manualmente os recursos dependentes do aplicativo (como fluxos de CloudWatch registros, funções do IAM e políticas do IAM).

    Ao criar seu aplicativo usando o CLI, você especifica qual versão do Apache Flink seu aplicativo usa usando o parâmetro `RuntimeEnvironment` da ação `CreateApplication`.

    Para obter um tutorial sobre como usar o CLI para criar um aplicativo, consulte the section called “Criar e executar um aplicativo usando a CLI” no Introdução (DataStream API) tutorial.

> **Note**
>
> Não é possível alterar o `RuntimeEnvironment` de um aplicativo existente. Se você precisar alterar o `RuntimeEnvironment` de um aplicativo existente, exclua o aplicativo e crie-o novamente.

Iniciar seu aplicativo Managed Service for Apache Flink

Depois de criar o código do aplicativo, carregá-lo no S3 e criar seu aplicativo Managed Service for Apache Flink, você inicia o aplicativo. O início de um aplicativo Managed Service for Apache Flink normalmente leva vários minutos.

Use um dos métodos a seguir para iniciar o aplicativo:

- Inicie seu aplicativo Managed Service for Apache Flink usando o console AWS: você pode executar seu aplicativo escolhendo Executar na página do seu aplicativo no console AWS.
- Inicie seu serviço gerenciado para o aplicativo Apache Flink usando a AWS API: você pode executar seu aplicativo usando a StartApplicationação.

Verificar seu aplicativo Managed Service for Apache Flink

Você pode verificar se o aplicativo está funcionando das seguintes maneiras:

- Usando CloudWatch registros: você pode usar o CloudWatch Logs e o CloudWatch Logs Insights para verificar se o aplicativo está funcionando corretamente. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo Managed Service for Apache Flink, consulte. Registro e Monitoramento
- Usando CloudWatch métricas: você pode usar CloudWatch métricas para monitorar a atividade do seu aplicativo ou a atividade nos recursos que seu aplicativo usa para entrada ou saída (como Kinesis Streams, Kinesis Data Firehose Streams ou buckets do Amazon S3). Para obter mais informações sobre CloudWatch métricas, consulte Como trabalhar com métricas no Guia CloudWatch do usuário da Amazon.
- Monitoramento de locais de saída: se seu aplicativo grava a saída em um local (como um bucket ou banco de dados do Amazon S3), você pode monitorar esse local para dados gravados.

Execução do aplicativo Managed Service for Apache Flink

Este tópico contém informações sobre como executar o Managed Service for Apache Flink.

Quando você executa o seu Managed Service for Apache Flink, o serviço cria um trabalho do Apache Flink. Um trabalho do Apache Flink é o ciclo de vida de execução do seu aplicativo Managed Service for Apache Flink. A execução do trabalho e os recursos que ele usa são gerenciados pelo Job Manager. O Job Manager divide a execução do aplicativo em tarefas. Cada tarefa é gerenciada por um gerenciador de tarefas. Ao monitorar o desempenho do seu aplicativo, você pode examinar o desempenho de cada gerenciador de tarefas ou do Job Manager como um todo.

Para obter informações sobre os trabalhos do Apache Flink, consulte Trabalhos e programação na documentação do Apache Flink.

Aplicativo e status do trabalho

Tanto seu aplicativo quanto o trabalho do aplicativo têm um status de execução atual:

- Status do aplicativo: seu aplicativo tem um status atual que descreve a sua fase de execução. Os status do aplicativo incluem o seguinte:
  - Status de aplicativo estacionário: seu aplicativo normalmente permanece nesses status até que você faça uma alteração de status:
    - READY: Um aplicativo novo ou interrompido está no status PRONTO até que você o execute.
    - RUNNING: Um aplicativo que foi iniciado com sucesso está no status EXECUTANDO.
  - Status transitórios de aplicativos: Um aplicativo nesses status normalmente está em processo de transição para outro status. Se um aplicativo permanecer em um status transitório por um período de tempo, você poderá interromper o aplicativo usando a ação StopApplication com o `Force` parâmetro definido como `true`. Esses status incluem o seguinte:
    - `STARTING:` Ocorre após a ação StartApplication. O aplicativo está passando do status `READY` para `RUNNING`.
    - `STOPPING:` Ocorre após a ação StopApplication. O aplicativo está passando do status `RUNNING` para `READY`.
    - `DELETING:`Ocorre após a ação DeleteApplication. O aplicativo está em processo de ser excluído.
    - `UPDATING:`Ocorre após a ação UpdateApplication. O aplicativo está sendo atualizado e voltará ao status `RUNNING` ou `READY`.

- `AUTOSCALING`:O aplicativo tem a propriedade `AutoScalingEnabled` do ParallelismConfiguration definida como `true`, e o serviço está aumentando o paralelismo do aplicativo. Quando o aplicativo está nesse status, a única ação de API válida que você pode usar é a ação StopApplication com o parâmetro `Force` definido como `true`. Para obter mais informações sobre escalabilidade, consulte Escalabilidade automática.
- `FORCE_STOPPING`: Ocorre depois que a ação StopApplication é chamada com o `Force` parâmetro definido como `true`. O aplicativo está em processo de ser interrompido à força. O aplicativo faz a transição do status `STARTING`, `UPDATING`, `STOPPING`, ou `AUTOSCALING` para o status`READY`.
- `ROLLING_BACK`: Ocorre depois que a ação RollbackApplication é chamada. O aplicativo está em processo de ser revertido para uma versão anterior. O aplicativo faz a transição do status `UPDATING` ou `AUTOSCALING` para o status `RUNNING`.
- `ROLLED_BACK`: Quando você reverte com sucesso um aplicativo, esse se torna o status da versão da qual você reverteu. Para obter informações sobre como reverter um aplicativo, consulte RollbackApplication.
- `MAINTENANCE`: Ocorre enquanto o Managed Service for Apache Flink aplica patches ao seu aplicativo. Para obter mais informações, consulte Manutenção.

Você pode verificar o status do seu aplicativo usando o console ou usando a ação DescribeApplication.

- Status do trabalho: quando seu aplicativo está no status `RUNNING`, o seu trabalho tem um status que descreve a fase de execução atual. Um trabalho começa no status `CREATED` e, em seguida, passa para o status `RUNNING` quando é iniciado. Se ocorrerem condições de erro, o seu aplicativo entrará no seguinte status:
  - Para aplicativos que usam o Apache Flink 1.11 e versões posteriores, seu aplicativo entra no status `RESTARTING`.
  - Para aplicativos que usam o Apache Flink 1.8 e versões anteriores, seu aplicativo entra no status `FAILING`.

  Em seguida, o aplicativo passa para o status `RESTARTING` ou `FAILED`, dependendo se o trabalho pode ser reiniciado.

  Você pode verificar o status do trabalho examinando o log do CloudWatch do seu aplicativo em busca de alterações de status.

Workloads em batch

O Managed Service for Apache Flink suporta a execução de workloads em batch do Apache Flink. Em um trabalho em lote, quando um trabalho do Apache Flink atinge o status FINALIZADO, o status do aplicativo Managed Service for Apache Flink é definido como PRONTO. Para obter mais informações sobre os status de trabalho do Flink, consulte Trabalhos e programação.

Recursos do aplicativo

Esta seção descreve os recursos do sistema que seu aplicativo usa. Entender como o Managed Service for Apache Flink provisiona e usa recursos ajudará você a projetar, criar e manter um Managed Service estável e com desempenho para o aplicativo Apache Flink.

Recursos do aplicativo do Managed Service for Apache Flink

O Managed Service for Apache Flink é um serviço AWS que cria um ambiente para hospedar seu aplicativo Apache Flink. O serviço do Managed Service for Apache Flink fornece recursos usando unidades chamadas Kinesis Processing Units (KPUs).

Uma KPU representa os seguintes recursos do sistema:

- Um Núcleo de CPU
- 4 GB de memória, dos quais um GB é memória nativa e três GB são memória heap
- 50 GB de espaço em disco

As KPUs executam aplicativos em unidades de execução distintas chamadas tarefas e subtarefas. Você pode pensar em uma subtarefa como o equivalente a um thread.

O número de KPUs disponíveis para um aplicativo é igual à configuração `Parallelism` do aplicativo, dividido pela configuração `ParallelismPerKPU` do aplicativo.

Para obter mais informações sobre paralelismo de aplicativo, consulte Escalabilidade.

Recursos do aplicativo Apache Flink

O ambiente Apache Flink aloca recursos para seu aplicativo usando unidades chamadas slots de tarefas. Quando o Managed Service for Apache Flink aloca recursos para seu aplicativo, ele atribui

um ou mais slots de tarefas do Apache Flink a uma única KPU. O número de slots atribuídos a uma única KPU é igual à configuração `ParallelismPerKPU` do seu aplicativo. Para obter mais informações sobre slots de tarefas, consulte Programação de trabalho na documentação do Apache Flink.

Paralelismo do operador

Você pode definir o número máximo de subtarefas que um operador pode usar. Esse valor é chamado de Paralelismo do operador. Por padrão, o paralelismo de cada operador em seu aplicativo é igual ao paralelismo do aplicativo. Isso significa que, por padrão, cada operador em seu aplicativo pode usar todas as subtarefas disponíveis no aplicativo, se necessário.

Você pode definir o paralelismo dos operadores em seu aplicativo usando o método `setParallelism`. Usando esse método, você pode controlar o número de subtarefas que cada operador pode usar ao mesmo tempo.

Para obter mais informações sobre o encadeamento de operadores, consulte Encadeamento de tarefas e grupos de recursos na Documentação do Apache Flink.

Encadeamento de operadores

Normalmente, cada operador usa uma subtarefa separada para executar, mas se vários operadores sempre forem executados em sequência, o runtime poderá atribuir todos à mesma tarefa. Esse processo é chamado de Encadeamento de operadores.

Vários operadores sequenciais podem ser encadeados em uma única tarefa se todos operarem com os mesmos dados. A seguir encontram-se alguns dos critérios necessários para que isso ocorra:

- Os operadores fazem um encaminhamento simples de 1 para 1.
- Todos os operadores têm o mesmo paralelismo de operadores.

Quando seu aplicativo encadeia operadores em uma única subtarefa, ele conserva os recursos do sistema, porque o serviço não precisa realizar operações de rede e alocar subtarefas para cada operador. Para determinar se seu aplicativo está usando o encadeamento de operadores, veja o gráfico de tarefas no console do Managed Service for Apache Flink. Cada vértice no aplicativo representa um ou mais operadores. O gráfico mostra operadores que foram encadeados como um único vértice.

API de fluxo de dados

Seu aplicativo Apache Flink usa a API Apache Flink DataStream para transformar dados em um fluxo de dados.

Esta seção contém os seguintes tópicos:

- Usando conectores para mover dados no serviço gerenciado para Apache Flink com a API DataStream : esses componentes movem dados entre seu aplicativo e fontes de dados e destinos externos.
- Transformar dados usando operadores no Managed Service for Apache Flink com a API DataStream: esses componentes transformam ou agrupam elementos de dados em seu aplicativo.
- Rastrear eventos no Managed Service for Apache Flink usando a API DataStream: este tópico descreve como o Managed Service for Apache Flink rastreia eventos ao usar a API DataStream.

Usando conectores para mover dados no serviço gerenciado para Apache Flink com a API DataStream

Na DataStream API Amazon Managed Service for Apache Flink, conectores são componentes de software que movem dados para dentro e para fora de um aplicativo Managed Service for Apache Flink. Os conectores são integrações flexíveis que permitem a leitura de arquivos e diretórios. Os conectores consistem em módulos completos para interagir com os serviços da Amazon e sistemas de terceiros.

Os tipos de conectores incluem o seguinte:

- Origens: forneça dados para seu aplicativo a partir de um fluxo de dados do Kinesis, arquivo ou de outra fonte de dados.
- Coletores: envie dados do seu aplicativo para um fluxo de dados do Kinesis, fluxo do Kinesis Data Firehose ou outro destino de dados.
- E/S assíncrona: fornece acesso assíncrono a uma fonte de dados (como um banco de dados) para enriquecer os eventos de fluxo.

Disponibilidade de conectores

A estrutura do Apache Flink contém conectores para acessar dados de várias fontes. Para obter informações sobre conectores disponíveis na estrutura do Apache Flink, consulte Conectores na Documentação do Apache Flink.

Warning

Se você tem aplicativos em execução no Flink 1.6, 1.8, 1.11 ou 1.13 e gostaria de executar nas regiões do Oriente Médio (EAU), Ásia-Pacífico (Hyderabad), Israel (Tel Aviv), Europa (Zurique), Oriente Médio (EAU), Ásia-Pacífico (Melbourne) e Ásia-Pacífico (Jacarta), talvez seja necessário reconstruir seu arquivo de aplicativos com um conector atualizado ou fazer o upgrade para o Flink 1.15. A seguir estão as diretrizes recomendadas:

Atualizações de conectores

V d Fl	Conector usado	Resolução
1, - 1,	Firehose	Seu aplicativo depende de uma versão desatualizada do conector Firehose que não conhece as regiões AWS

V d Fl	Conector usado	Resolução
		mais recentes. Reconstrua o arquivo do seu aplicativo com o conector Firehose versão 2.1.0. v2.1.0

V d Fl	Conector usado	Resolução
1.	Kinesis	Seu aplicativo depende de uma versão desatualizada do conector Flink Kinesis que não conhece as regiões AWS mais recentes. Reconstrua o arquivo do seu aplicativo com o conector

V d F	Conector usado	Resolução
		Flink Kinesis versão 1.6.1. https://github.com/awslabs/amazon-kinesis-connector-flink/árvore/1.6.1

V d Fl	Conector usado	Resolução
1.	Kinesis	Seu aplicativo depende de uma versão desatualizada do conector Flink Kinesis que não conhece as regiões AWS mais recentes. Reconstrua o arquivo do seu aplicativo com o conector

V d Fl	Conector usado	Resolução
		Flink Kinesis versão 2.4.1. https://github.com/awslabs/amazon-kinesis-connector-flink/árvore/2.4.1

V d Fl	Conector usado	Resolução
1. e 1.	Kinesis	Seu aplicativo depende de uma versão desatualizada do conector Flink Kinesis que não conhece as regiões AWS mais recentes. Infelizmente, o Flink não lança mais patches ou correções de

V d Fl	Conector usado	Resolução
		bugs para conectores 1.6/1.13. Sugerimos que você atualize para o Flink 1.15 reconstruindo seu arquivo de aplicativos com o Flink 1.15.

Adicionar fontes de dados de transmissão ao Managed Service for Apache Flink

O Apache Flink fornece conectores para leitura de arquivos, soquetes, coleções e fontes personalizadas. No código do seu aplicativo, você usa uma fonte do Apache Flink para receber dados de um fluxo. Esta seção descreve as fontes que estão disponíveis para os serviços da Amazon.

Kinesis Data Streams

A fonte `FlinkKinesisConsumer` fornece dados de transmissão para seu aplicativo a partir de um fluxo de dados da Amazon Kinesis.

Criação de um **`FlinkKinesisConsumer`**

O exemplo de código a seguir demonstra como criar um `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 obter mais informações sobre como usar uma `FlinkKinesisConsumer`, consulte Baixar e examinar o código Java Apache Flink Streaming.

Criação de um **`FlinkKinesisConsumer`** que usa um consumidor EFO

FlinkKinesisConsumer Agora é compatível com Enhanced Fan-Out (EFO).

Se um consumidor do Kinesis usa o EFO, o serviço Kinesis Data Streams fornece sua própria largura de banda dedicada, em vez de fazer com que o consumidor compartilhe a largura de banda fixa do stream com os outros consumidores que estão lendo o stream.

Para obter mais informações sobre como usar o EFO com o consumidor Kinesis, consulte FLIP-128: distribuição avançada para consumidores da AWS Kinesis.

Você habilita o consumidor EFO definindo os seguintes parâmetros no consumidor do Kinesis:

- RECORD_PUBLISHER_TYPE: defina esse parâmetro como EFO para que seu aplicativo use um consumidor EFO para acessar os dados do Kinesis Data Stream.
- EFO_CONSUMER_NAME: defina esse parâmetro como um valor de sequência de caracteres que é exclusivo entre os consumidores desse fluxo. A reutilização de um nome de consumidor no mesmo Kinesis Data Stream fará com que o consumidor anterior que usava esse nome seja excluído.

Para configurar um `FlinkKinesisConsumer` para usar o EFO, adicione os seguintes parâmetros ao consumidor:

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

Para obter um exemplo de um aplicativo do Managed Service for Apache Flink que usa um consumidor EFO, consulte Consumidor EFO.

Amazon MSK

A fonte `KafkaSource` fornece dados de transmissão para seu aplicativo a partir de um tópico do Amazon MSK.

Criação de uma **`KafkaSource`**

O exemplo de código a seguir demonstra como criar um `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 obter mais informações sobre como usar uma `KafkaSource`, consulte Replicação do MSK.

Gravar dados com coletores no Managed Service for Apache Flink

No código do seu aplicativo, você usa um coletor do Apache Flink para gravar dados de um fluxo do Apache Flink em um serviço AWS, como o Kinesis Data Streams.

O Apache Flink fornece coletores para arquivos, soquetes e coletores personalizados. Os seguintes coletores estão disponíveis para o AWS:

Kinesis Data Streams

O Apache Flink fornece informações sobre o conector do Kinesis Data Streams na documentação do Apache Flink.

Para obter um exemplo de um aplicativo que usa um fluxo de dados Kinesis para entrada e saída, consulte Introdução (DataStream API).

Amazon S3

É possível utilizar o `StreamingFileSink` do Apache Flink para gravar objetos em um bucket do Amazon S3.

Para obter um exemplo sobre como gravar objetos no S3, consulte the section called “Coletor do S3”.

Kinesis Data Firehose

O `FlinkKinesisFirehoseProducer` é um coletor do Apache Flink confiável e escalável para armazenar a saída do aplicativo usando o serviço Kinesis Data Firehose. Esta seção descreve como configurar um projeto do Maven para criar e utilizar um `FlinkKinesisFirehoseProducer`.

Tópicos

Criação de um **`FlinkKinesisFirehoseProducer`**

O exemplo de código a seguir demonstra como criar um `FlinkKinesisFirehoseProducer`:

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

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

Exemplo de código **`FlinkKinesisFirehoseProducer`**

O exemplo de código a seguir demonstra como criar, configurar `FlinkKinesisFirehoseProducer` e enviar dados de um fluxo de dados do Apache Flink para o serviço 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 um tutorial completo sobre como usar o coletor Kinesis Data Firehose, consulte the section called “Coletor do Kinesis Data Firehose”.

Usar E/S assíncrona no Managed Service for Apache Flink

Um operador de E/S assíncrona enriquece os dados de fluxo usando uma fonte de dados externa, como um banco de dados. O Managed Service for Apache Flink enriquece os eventos de transmissão de forma assíncrona para que as solicitações possam ser agrupadas em lotes para maior eficiência.

Para obter mais informações, consulte E/S assíncrona na Documentação do Apache Flink.

Transformar dados usando operadores no Managed Service for Apache Flink com a API DataStream

Para transformar os dados recebidos em um Managed Service for Apache Flink, você usa um operador do Apache Flink. Um operador do Apache Flink transforma um ou mais fluxos de dados em um novo fluxo de dados. O novo fluxo de dados contém dados modificados do fluxo de dados original. O Apache Flink fornece mais de 25 operadores de processamento de fluxo pré-definidos. Para obter mais informações, consulte Operadores na Documentação do Apache Flink.

Este tópico contém as seguintes seções:

- Operadores de transformação
- Operadores de agregação

Operadores de transformação

Veja a seguir um exemplo de uma transformação de texto simples em um dos campos de um fluxo de dados JSON.

Esse código cria um fluxo de dados transformado. O novo fluxo de dados tem os mesmos dados do fluxo original, com a string " `Company`" anexada ao conteúdo do 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 agregação

Este é um exemplo de um operador de agregação. O código cria um fluxo de dados agregado. O operador cria uma janela em cascata de 5 segundos e retorna a soma dos valores de `PRICE` dos registros na janela com o mesmo valor de `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 obter um exemplo de código completo que usa operadores, consulte Introdução (DataStream API). O código-fonte dos conceitos básicos do aplicativo está disponível em Conceitos básicos no repositório GitHub dos Exemplos de Java do Managed Service for Apache Flink.

Rastrear eventos no Managed Service for Apache Flink usando a API DataStream

O Managed Service for Apache Flink rastreia eventos usando os seguintes timestamps:

- Tempo de processamento: refere-se à hora do sistema da máquina que está executando a respectiva operação.
- Hora do evento: refere-se à hora em que cada evento individual ocorreu em seu dispositivo produtor.
- Tempo de ingestão: refere-se à hora em que os eventos entram no serviço Managed Service for Apache Flink.

Você define o tempo usado pelo ambiente de transmissão usando `setStreamTimeCharacteristic`:

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

Para obter mais informações sobre timestamps, consulte Hora do evento em Documentação do Apache Flink.

API de tabela

Seu aplicativo Apache Flink usa a API de tabela do Apache Flink para interagir com dados em um fluxo usando um modelo relacional. Você usa a API de tabela para acessar dados usando fontes de tabela e, em seguida, usa funções de tabela para transformar e filtrar dados da tabela. Você pode transformar e filtrar dados tabulares usando funções de API ou comandos SQL.

Esta seção contém os seguintes tópicos:

- Conectores de API de tabela: esses componentes movem dados entre seu aplicativo e fontes de dados e destinos externos.
- Atributos de tempo da API de tabela: este tópico descreve como o Managed Service for Apache Flink rastreia eventos ao usar a API de tabela.

Conectores de API de tabela

No modelo de programação do Apache Flink, os conectores são componentes que seu aplicativo usa para ler ou gravar dados de fontes externas, como outros serviços de AWS.

Com a API de tabela do Apache Flink, você pode usar os seguintes tipos de conectores:

- Fontes da API de tabela: use conectores de origem da API de tabela para criar tabelas dentro da sua `TableEnvironment` usando chamadas de API ou consultas SQL.
- Coletores de API de tabela: use comandos SQL para gravar dados de tabela em fontes externas, como um tópico do Amazon MSK ou um bucket do Amazon S3.

Fontes da API de tabela

Você cria uma fonte de tabela a partir de um fluxo de dados. O código a seguir cria uma tabela a partir de um tópico do 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 obter mais informações sobre fontes de tabelas, consulte Tabela e conectores na Documentação do Apache Flink.

Coletores de API de tabela

Para gravar dados da tabela em um coletor, você cria o coletor em SQL e, em seguida, executa o coletor baseado em SQL no objeto `StreamTableEnvironment`.

O exemplo de código a seguir demonstra como gravar dados de tabela em um coletor do 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");
```

Você pode usar o parâmetro `format` para controlar qual formato o Managed Service for Apache Flink usa para gravar a saída no coletor. Para obter informações sobre formatos, consulte Formatos na Documentação do Apache Flink.

Para obter mais informações sobre coletores de tabelas, consulte Tabela e conectores na Documentação do Apache Flink.

Fontes e coletores definidos pelo usuário

Você pode usar os conectores Apache Kafka existentes para enviar dados de e para outros serviços de AWS, como Amazon MSK e Amazon S3. Para interagir com outras fontes de dados e destinos, você pode definir suas próprias fontes e coletores. Para obter mais informações, consulte Fontes e coletores definidos pelo usuário na Documentação do Apache Flink.

Atributos de tempo da API de tabela

Cada registro em um fluxo de dados tem vários timestamps que definem quando os eventos relacionados ao registro ocorreram:

- Hora do evento: um timestamp definido pelo usuário que define quando o evento que criou o registro ocorreu.
- Tempo de ingestão: o momento em que seu aplicativo recuperou o registro do fluxo de dados.
- Tempo de processamento: o momento em que seu aplicativo processou o registro.

Quando a API de tabela do Apache Flink cria janelas com base nas horas registradas, você define quais desses timestamps ela usa usando o método setStreamTimeCharacteristic.

Para obter mais informações sobre o uso de timestamps com a API de tabela, consulte Atributos de tempo na Documentação do Apache Flink.

Usar o Python com Managed Service for Apache Flink

> **Note**
>
> Se você estiver desenvolvendo um aplicativo Python Flink em um novo Mac com o chip Apple Silicon, poderá encontrar alguns problemas conhecidos com as dependências do Python do PyFlink 1.15. Nesse caso, recomendamos executar o interpretador Python no Docker. Para obter as instruções do passo a passo, consulte Desenvolvimento do PyFlink 1.15 no Apple Silicon Mac.

O Apache Flink versão 1.15.2 inclui suporte para criar aplicativos usando o Python versão 3.8, usando a biblioteca PyFlink. Você cria um serviço gerenciado para o aplicativo Apache Flink usando Python fazendo o seguinte:

- Crie o código do seu aplicativo Python como um arquivo de texto com um `main` método.
- Empacote o arquivo de código do seu aplicativo e todas as dependências do Python ou Java em um arquivo zip e faça o upload para um bucket do Amazon S3.
- Crie seu serviço gerenciado para o aplicativo Apache Flink, especificando a localização do código do Amazon S3, as propriedades do aplicativo e as configurações do aplicativo.

Em um alto nível, a Python Table API é um invólucro em torno da API Java Table. Para obter informações sobre a API Python Table, consulte Introdução à API Python Table na documentação do Apache Flink.

Programação do seu serviço gerenciado for Apache Flink for Python

Você codifica seu aplicativo Python para o Managed Service for Apache Flink usando a API Apache Flink Python Table. O mecanismo Apache Flink traduz as instruções da Python Table API (em execução na Python VM) em instruções da Java Table API (executadas na Java VM).

Para isso, você usa a Python Table API da seguinte maneira:

- Crie uma referência para o `StreamTableEnvironment`.
- Crie `table` objetos a partir dos dados de streaming de origem executando consultas na `StreamTableEnvironment` referência.
- Execute consultas em seus `table` objetos para criar tabelas de saída.

- Grave suas tabelas de saída em seus destinos usando um`StatementSet`.

Para começar a usar a Python Table API no Managed Service for Apache Flink, consulte. Conceitos básicos do Amazon Managed Service for Apache Flink para Python

Ler e gravar dados de streaming

Para ler e gravar dados de streaming, você executa consultas SQL no ambiente de tabela.

Criar uma tabela

O exemplo de código a seguir demonstra uma função definida pelo usuário que cria uma consulta SQL. A consulta SQL cria uma tabela que interage com um stream do 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)
```

Ler dados de streaming

O exemplo de código a seguir demonstra como usar a consulta `CreateTable` SQL anterior em uma referência de ambiente de tabela para ler dados:

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

Escrever dados de streaming

O exemplo de código a seguir demonstra como usar a consulta SQL do `CreateTable` exemplo para criar uma referência de tabela de saída e como usar a `StatementSet` para interagir com as tabelas para escrever dados em um stream do Kinesis de destino:

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

Ler propriedades de runtime

Você pode usar propriedades de runtime para configurar seu aplicativo sem alterar o código do aplicativo.

Você especifica as propriedades do aplicativo da mesma forma que com um Managed Service for Apache Flink para aplicativo Java. É possível especificar as propriedades do runtime das seguintes maneiras:

- Usar a ação CreateApplication.
- Usar a ação UpdateApplication.
- Usar seu aplicativo usando o console.

Você recupera as propriedades do aplicativo no código lendo um arquivo json chamado `application_properties.json` aquele criado pelo runtime do Managed Service for Apache Flink.

O exemplo de código a seguir demonstra a leitura das propriedades do aplicativo a partir do arquivo `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)
```

O exemplo de código de função definido pelo usuário a seguir demonstra a leitura de um grupo de propriedades do objeto de propriedades do aplicativo: recupera:

```
def property_map(properties, property_group_id):
```

```
    for prop in props:
        if prop["PropertyGroupId"] == property_group_id:
            return prop["PropertyMap"]
```

O exemplo de código a seguir demonstra a leitura de uma propriedade chamada INPUT_STREAM_KEY de um grupo de propriedades que o exemplo anterior retorna:

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

Criar o pacote de código do seu aplicativo

Depois de criar seu aplicativo Python, você empacota seu arquivo de código e dependências em um arquivo zip.

Seu arquivo zip deve conter um script python com um método `main` e, opcionalmente, pode conter o seguinte:

- Arquivos de código Python adicionais
- Código Java definido pelo usuário em arquivos JAR
- Bibliotecas Java em arquivos JAR

> **Note**
>
> O arquivo zip do aplicativo deve conter todas as dependências do aplicativo. Você não pode referenciar bibliotecas de outras fontes para seu aplicativo.

Criar seu aplicativo Managed Service for Apache Flink Python

Especificar seus arquivos de código

Quando você tiver criado o pacote de código do seu aplicativo, você deve carregá-lo em um bucket do Amazon S3. Em seguida, crie um aplicativo usando o console ou a ação CreateApplication.

Ao criar seu aplicativo usando a ação CreateApplication, você especifica os arquivos de código e arquivamentos em seu arquivo zip usando um grupo especial de propriedades do aplicativo chamado `kinesis.analytics.flink.run.options`. Você pode definir os seguintes tipos de arquivos:

- python: um arquivo de texto contendo um método principal do Python.

- jarfile: um arquivo Java JAR contendo funções Java definidas pelo usuário.
- pyFiles: um arquivo de recursos do Python contendo recursos a serem usados pelo aplicativo.
- pyArchives: um arquivo zip contendo arquivos de recursos para o aplicativo.

Para obter mais informações sobre os tipos de arquivo de código do Apache Flink Python, consulte Uso da linha de comando na documentação do Apache Flink.

> **Note**
>
> O Managed Service for Apache Flink não suporta os tipos de arquivo `pyModule`, `pyExecutable` ou `pyRequirements`. Todo o código, requisitos e dependências devem estar em seu arquivo zip. Você não pode especificar dependências a serem instaladas usando pip.

O exemplo de trecho json a seguir demonstra como especificar a localização dos arquivos no arquivo zip do seu aplicativo:

```
"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"
          }
        },
```

Monitoramento do seu aplicativo Python Managed Service for Apache Flink

Você usa o log do CloudWatch do seu aplicativo para monitorar seu aplicativo Managed Service for Apache Flink Python.

O Managed Service for Apache Flink registra as seguintes mensagens para aplicativos Python:

- Mensagens escritas no console usando o método do `print()`aplicativo`main`.

- Mensagens enviadas em funções definidas pelo usuário usando o pacote `logging`. O exemplo de código a seguir demonstra a gravação no log do aplicativo a partir de uma função definida pelo usuário:

```
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
```

- Mensagens de erro lançadas pelo aplicativo.

  Se o aplicativo gerar uma exceção na função `main`, ela aparecerá nos registros do seu aplicativo.

  O exemplo a seguir demonstra uma entrada de registro para uma exceção lançada a partir do 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**
>
> Devido a problemas de desempenho, recomendamos que você use somente mensagens de log personalizadas durante o desenvolvimento do aplicativo.

Consultar logs com o CloudWatch Insights

A seguinte consulta do CloudWatch Insights pesquisa logs criados pelo ponto de entrada do Python enquanto executa a função principal do seu aplicativo:

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

Propriedades de runtime no Managed Service for Apache Flink

Você pode usar propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.

Este tópico contém as seguintes seções:

- Trabalhar com propriedades de runtime no console
- Trabalhar com propriedades de runtime na CLI
- Acessar propriedades de runtime em um aplicativo Managed Service for Apache Flink

Trabalhar com propriedades de runtime no console

Você pode adicionar, atualizar ou remover propriedades de runtime do seu aplicativo Managed Service for Apache Flink usando o console.

> **Note**
>
> Você não pode adicionar propriedades de runtime ao criar um aplicativo no console do Managed Service for Apache Flink.

Atualizar propriedades de runtime for Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. Selecione o seu aplicativo Managed Service for Apache Flink. Selecione Detalhes do aplicativo.
3. Na página do seu aplicativo, selecione Configurar.
4. Expanda a seção Propriedades.
5. Use os controles na seção Propriedades para definir um grupo de propriedades com pares de valor-chave. Use esses controles para adicionar, atualizar ou remover grupos de propriedades e propriedades de runtime.
6. Selecione Atualizar.

Trabalhar com propriedades de runtime na CLI

É possível adicionar, atualizar ou remover propriedades de runtime usando AWS CLI.

Esta seção inclui exemplos de solicitações de ações de API para configurar propriedades de runtime para um aplicativo. Para obter informações sobre como usar um arquivo JSON como entrada para uma ação de API, consulte Exemplo de código de API para o Managed Service for Apache Flink.

> **Note**
>
> Substitua o exemplo de ID da conta (*012345678901*) nos exemplos a seguir pelo ID da sua conta.

Adicionar propriedades de runtime ao criar um aplicativo

O exemplo a seguir de solicitação para a ação `CreateApplication` adiciona dois grupos de propriedades de runtime (`ProducerConfigProperties` e `ConsumerConfigProperties`) quando você cria um aplicativo:

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

Adicionar e atualizar propriedades de runtime em um aplicativo existente

O exemplo a seguir de solicitação para a ação `UpdateApplication` adiciona ou atualiza as propriedades de runtime de um aplicativo 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**
>
> Se você usar uma chave que não tenha uma propriedade de runtime correspondente em um grupo de propriedades, o Managed Service for Apache Flink adicionará o par de chave-valor como uma nova propriedade. Se você usar uma chave para uma propriedade de runtime existente em um grupo de propriedades, o Managed Service for Apache Flink atualizará o valor da propriedade.

Remover propriedades de runtime

O exemplo a seguir de solicitação para a ação `UpdateApplication` remove todas as propriedades de runtime e grupos de propriedades de um aplicativo existente:

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

> **Important**
>
> Se você omitir um grupo de propriedades existente ou uma chave de propriedade existente em um grupo de propriedades, esse grupo de propriedades ou propriedade será removido.

Acessar propriedades de runtime em um aplicativo Managed Service for Apache Flink

Você recupera as propriedades de runtime no código do aplicativo Java usando o método estático `KinesisAnalyticsRuntime.getApplicationProperties()`, que retorna um objeto `Map<String, Properties>`.

O exemplo de código Java a seguir recupera as propriedades de runtime do seu aplicativo:

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

Você recupera um grupo de propriedades (como um objeto `Java.Util.Properties`) da seguinte forma:

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

Normalmente, você configura uma fonte ou coletor do Apache Flink passando no objeto `Properties` sem precisar recuperar as propriedades individuais. O exemplo de código a seguir demonstra como criar uma fonte do Flink transmitindo em um objeto `Properties` recuperado das propriedades de runtime:

```
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 obter um exemplo de código completo que usa propriedades de runtime, consulte Introdução (DataStream API). O código-fonte dos conceitos básicos do aplicativo está disponível em Conceitos básicos no repositório GitHub dos Exemplos de Java do Managed Service for Apache Flink.

Implementar tolerância a falhas no Managed Service for Apache Flink

O ponto de verificação é o método usado para implementar a tolerância a falhas no Amazon Managed Service for Apache Flink. Um ponto de verificação é um backup atualizado de um aplicativo em execução que é usado para se recuperar imediatamente de uma interrupção ou failover inesperado do aplicativo.

Para obter detalhes sobre pontos de verificação em aplicativos do Apache Flink, consulte Pontos de verificação na Documentação do Apache Flink.

Um snapshot é um backup do estado do aplicativo criado e gerenciado manualmente. Os snapshots permitem que você restaure seu aplicativo para um estado anterior chamando `UpdateApplication`. Para obter mais informações, consulte Gerenciar backups de aplicativos usando snapshots.

Se o ponto de verificação estiver habilitado para seu aplicativo, o serviço fornecerá tolerância a falhas criando e carregando backups dos dados do aplicativo no caso de reinicializações inesperadas do aplicativo. Essas reinicializações inesperadas de aplicativos podem ser causadas por reinicializações inesperadas de tarefas, falhas de instância etc. Isso dá ao aplicativo a mesma semântica da execução sem falhas durante essas reinicializações.

Se os snapshots forem habilitados para o aplicativo e configurados usando o ApplicationRestoreConfiguration do aplicativo, o serviço fornecerá uma semântica de processamento exatamente uma vez durante as atualizações do aplicativo ou durante a escalabilidade ou a manutenção relacionada ao serviço.

Configurar ponto de verificação no Managed Service for Apache Flink

Você pode configurar o comportamento de ponto de verificação do seu aplicativo. Você pode definir se ele persiste no estado de ponto de verificação, com que frequência ele salva seu estado nos pontos de verificação e o intervalo mínimo entre o final de uma operação de ponto de verificação e o início de outra.

Você define as seguintes configurações usando as operações de API `CreateApplication` ou `UpdateApplication`:

- `CheckpointingEnabled` — Indica se o ponto de verificação está ativado no aplicativo.
- `CheckpointInterval` — Contém o tempo em milissegundos entre as operações do ponto de verificação (persistência).
- `ConfigurationType` — Defina esse valor para `DEFAULT` para usar o comportamento do ponto de verificação padrão. Defina esse valor para `CUSTOM` para configurar outros valores.

> **Note**
>
> O comportamento padrão do ponto de verificação é o seguinte:
>
> - CheckpointingEnabled: true
> - CheckpointInterval: 60000
> - MinPauseBetweenCheckpoints: 5000
>
> Se ConfigurationType for definido como `DEFAULT`, os valores anteriores serão usados, mesmo se eles estiverem definidos de outra forma usando AWS Command Line Interface, ou definindo os valores no código do aplicativo.

> **Note**
>
> Para o Flink 1.15 em diante, o Managed Service for Apache Flink usará `stop-with-savepoint` durante a criação automática de instantâneos, ou seja, a atualização, a escalabilidade ou a parada do aplicativo.

- `MinPauseBetweenCheckpoints` — O tempo mínimo em milissegundos entre o final de uma operação de ponto de verificação e o início de outra. Definir esse valor impede o aplicativo de verificar continuamente quando uma operação de ponto de verificação levar mais tempo do que `CheckpointInterval`.

Exemplos de API de ponto de verificação

Esta seção inclui exemplos de solicitações de ações de API para configurar pontos de verificação para um aplicativo. Para obter informações sobre como usar um arquivo JSON como entrada de uma ação da API, consulte Exemplo de código de API para o Managed Service for Apache Flink.

Configurar o ponto de verificação para um novo aplicativo

O exemplo de solicitação a seguir para a ação `CreateApplication` configura o ponto de verificação quando você está criando um aplicativo:

```
{
   "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
         }
      }
}
```

Desativar o ponto de verificação para um novo aplicativo

O exemplo de solicitação a seguir para a ação `CreateApplication` desativa o ponto de verificação quando você está criando um aplicativo:

```
{
   "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 o ponto de verificação para um aplicativo existente

O exemplo de solicitação a seguir para a ação `UpdateApplication` configura o ponto de verificação para um aplicativo existente:

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

Desativar o ponto de verificação para um aplicativo existente

O exemplo de solicitação a seguir para a ação `UpdateApplication` desativa o ponto de verificação para um aplicativo existente:

```
{
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": false,
            "CheckpointIntervalUpdate": 20000,
```

```
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

Gerenciar backups de aplicativos usando snapshots

Um snapshot é a implementação Managed Service for Apache Flink de um Ponto de salvamento do Apache Flink. Um instantâneo é um backup do estado do aplicativo acionado, criado e gerenciado pelo usuário ou serviço. Para obter informações sobre os pontos de salvamento do Apache Flink, consulte Pontos de salvamento na Documentação do Apache Flink. Usando snapshots, você pode reiniciar um aplicativo a partir de um instantâneo específico do estado do aplicativo.

> **Note**
>
> Recomendamos que seu aplicativo crie um instantâneo várias vezes ao dia para reiniciar adequadamente com os dados de estado corretos. A frequência correta para seus instantâneos depende da lógica de negócios do seu aplicativo. Tirar snapshots com frequência permite recuperar dados mais recentes, mas aumenta os custos e exige mais recursos do sistema.

No Managed Service for Apache Flink, você gerencia snapshots usando as seguintes ações de API:

- `CreateApplicationSnapshot`
- `DeleteApplicationSnapshot`
- `DescribeApplicationSnapshot`
- `ListApplicationSnapshots`

Para saber o limite do número de snapshots por aplicativo, consulte Quota. Se seu aplicativo atingir o limite de snapshots, a criação manual de um snapshot falhará com um `LimitExceededException`.

O Managed Service for Apache Flink nunca exclui snapshots. Será necessário excluir manualmente seus snapshots usando a ação `DeleteApplicationSnapshot`.

Para carregar um snapshot salvo do estado do aplicativo ao iniciar um aplicativo, use o parâmetro `ApplicationRestoreConfiguration` da ação `StartApplication` ou `UpdateApplication`.

Este tópico contém as seguintes seções:

- Criação automática de snapshots
- Restaurar a partir de um snapshot que contém dados de estado incompatíveis
- Exemplos da API de snapshot

Criação automática de snapshots

Se `SnapshotsEnabled` estiver definido como `true` no ApplicationSnapshotConfiguration do aplicativo, o Managed Service for Apache Flink cria e usa automaticamente snapshots quando o aplicativo é atualizado, escalado ou interrompido para fornecer uma semântica de processamento exatamente uma vez.

> **Note**
>
> Definir `ApplicationSnapshotConfiguration::SnapshotsEnabled` como `false` levará à perda de dados durante as atualizações do aplicativo.

> **Note**
>
> O Managed Service for Apache Flink aciona pontos de salvamento intermediários durante a criação do snapshot. Para a versão 1.15 ou superior do Flink, os pontos de salvamento intermediários não causam mais efeitos secundários. Consulte Acionamento de pontos de salvamento

Os snapshots criados automaticamente têm as seguintes qualidades:

- O snapshot é gerenciado pelo serviço, mas você pode vê-lo usando a ação ListApplicationSnapshots. Os snapshots criados automaticamente são contabilizados no seu limite de snapshots.
- Se seu aplicativo exceder o limite de snapshots, os snapshots criados manualmente falharão, mas o serviço Managed Service for Apache Flink ainda criará snapshots com êxito quando o aplicativo

for atualizado, escalado ou interrompido. Você deve excluir manualmente os snapshots usando a ação DeleteApplicationSnapshot antes de criar mais instantâneos manualmente.

Restaurar a partir de um snapshot que contém dados de estado incompatíveis

Como os snapshots contêm informações sobre operadores, a restauração dos dados de estado de um snapshot para um operador que foi alterado desde a versão anterior do aplicativo pode ter resultados inesperados. Um aplicativo falhará se tentar restaurar dados de estado de um snapshot que não corresponda ao operador atual. O aplicativo com falha ficará preso no estado `STOPPING` ou `UPDATING`.

Para permitir que um aplicativo restaure a partir de um snapshot que contém dados de estado incompatíveis, defina o parâmetro `AllowNonRestoredState` da FlinkRunConfiguration para `true` usando a ação UpdateApplication.

Você verá o seguinte comportamento quando um aplicativo for restaurado a partir de um snapshot obsoleto:

- Operador adicionado: se um novo operador for adicionado, o ponto de salvamento não terá dados de estado para o novo operador. Nenhuma falha ocorrerá e não é necessário configurar `AllowNonRestoredState`.
- Operador excluído: se um operador existente for excluído, o ponto de salvamento terá dados de estado do operador ausente. Ocorrerá uma falha, a menos que `AllowNonRestoredState` esteja configurada como`true`.
- Operador modificado: se forem feitas alterações compatíveis, como alterar o tipo de um parâmetro para um tipo compatível, o aplicativo poderá restaurar a partir do snapshot obsoleto. Para obter mais informações sobre restauração a partir de snapshot, consulte Pontos de salvamento na Documentação do Apache Flink. Um aplicativo que usa o Apache Flink versão 1.8 ou posterior pode ser restaurado a partir de um snapshot com um esquema diferente. Um aplicativo que usa o Apache Flink versão 1.6 não pode ser restaurado. Para coletores de confirmação em duas fases, recomendamos usar o snapshot do sistema (SWs) em vez do snapshot criado pelo usuário (CreateApplicationSnapshot).

  Para Flink, o Managed Service for Apache Flink aciona pontos de salvamento intermediários durante a criação do snapshot. Para o Flink 1.15 ou superior, os pontos de salvamento intermediários não causam mais efeitos secundários. Consulte Acionamento de pontos de salvamento.

Se você precisar retomar um aplicativo incompatível com os dados existentes do ponto de salvamento, recomendamos que você ignore a restauração a partir do instantâneo definindo o parâmetro `ApplicationRestoreType` da ação StartApplication como `SKIP_RESTORE_FROM_SNAPSHOT`.

Para obter mais informações sobre como o Apache Flink lida com dados de estado incompatíveis, consulte Evolução do esquema do estado na Documentação do Apache Flink.

Exemplos da API de snapshot

Esta seção inclui exemplos de solicitações de ações de API para usar snapshot com um aplicativo. Para obter informações sobre como usar um arquivo JSON como entrada de uma ação da API, consulte Exemplo de código de API para o Managed Service for Apache Flink.

Habilitar snapshots para um aplicativo

O exemplo de solicitação a seguir para a ação `UpdateApplication` habilita snapshots para um aplicativo:

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

Criar um snapshot

O exemplo de solicitação da ação `CreateApplicationSnapshot` a seguir cria um snapshot do estado atual do aplicativo:

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

Listar snapshots de um aplicativo

O exemplo de solicitação da ação `ListApplicationSnapshots` a seguir lista os primeiros 50 snapshots do estado atual do aplicativo:

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

Listar detalhes de um snapshot de aplicativo

O exemplo de solicitação a seguir para a ação `DescribeApplicationSnapshot` lista detalhes de um snapshot de aplicativo específico:

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

Excluir um snapshot

O exemplo de solicitação da ação `DeleteApplicationSnapshot` a seguir exclui um snapshot salvo anteriormente. Você pode obter o valor `SnapshotCreationTimestamp` usando um `ListApplicationSnapshots` ou `DeleteApplicationSnapshot`:

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

Reiniciar um aplicativo usando um snapshot nomeado

O exemplo a seguir de solicitação para a ação `StartApplication` inicia o aplicativo usando o estado salvo de um snapshot específico:

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_CUSTOM_SNAPSHOT",
```

```
            "SnapshotName": "MyCustomSnapshot"
         }
      }
}
```

Reiniciar um aplicativo usando o snapshot mais recente

O exemplo de solicitação para a ação `StartApplication` a seguir inicia o aplicativo usando o snapshot mais recente:

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

Reiniciar um aplicativo sem snapshot

O exemplo de solicitação para a ação `StartApplication` a seguir inicia o aplicativo sem carregar o estado do aplicativo, mesmo que um snapshot esteja presente:

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

Escalabilidade de aplicativos no Managed Service for Apache Flink

Você pode configurar a execução paralela de tarefas e a alocação de recursos para que o Amazon Managed Service for Apache Flink implemente a escalabilidade. Para obter mais informações sobre como o Apache Flink programa instâncias paralelas de tarefas, consulte Execução paralela na Documentação do Apache Flink.

Tópicos

Configurando o paralelismo de aplicativos e a KPU ParallelismPer

Você configura a execução paralela das tarefas do aplicativo Managed Service for Apache Flink (como ler de uma fonte ou executar um operador) usando as seguintes `ParallelismConfiguration` propriedades:

- `Parallelism` — Use esta propriedade para definir o paralelismo padrão do aplicativo Apache Flink. Todos os operadores, fontes e coletores são executados com esse paralelismo, a menos que sejam substituídos no código do aplicativo. O valor padrão é 1 e o máximo padrão é 256.
- `ParallelismPerKPU` — Use esta propriedade para definir o número de tarefas em paralelo que podem ser programadas por unidade de processamento do Kinesis (Kinesis Processing Unit, KPU) do seu aplicativo. O padrão é 1 e o máximo é 8. Para aplicativos que têm operações de bloqueio (por exemplo, E/S), um valor maior que `ParallelismPerKPU` leva à utilização total dos recursos da KPU.

> **Note**
>
> O limite para `Parallelism` é igual a `ParallelismPerKPU` vezes o limite para KPUs (que tem um padrão de 64). O limite de KPUs pode ser aumentado ao solicitar um aumento de limite. Para obter instruções sobre como solicitar um aumento de limite, consulte “Para solicitar um aumento de limite” em Service Quotas.

Para obter informações sobre como definir o paralelismo de tarefas para um operador específico, consulte Configurar o paralelismo: operador na documentação do Apache Flink.

Alocação de unidades de processamento do Kinesis

O Managed Service for Apache Flink provisiona a capacidade como KPUs. Uma única KPU oferece 1 vCPU e 4 GB de memória. Para cada KPU alocada, também são fornecidos 50 GB de armazenamento de aplicativos em execução.

O Managed Service for Apache Flink calcula as KPUs necessárias para executar seu aplicativo usando as propriedades `Parallelism` e `ParallelismPerKPU`, da seguinte forma:

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

O Managed Service for Apache Flink fornece rapidamente aos seus aplicativos recursos em resposta a picos no throughput ou na atividade de processamento. Ele remove recursos do seu aplicativo gradualmente após o pico de atividade ter passado. Para desativar a alocação automática de recursos, defina o valor `AutoScalingEnabled` como `false`, conforme descrito posteriormente em Atualizar o paralelismo do seu aplicativo.

O limite padrão para KPUs para seu aplicativo é 64. Para obter instruções sobre como solicitar um aumento desse limite, consulte “Para solicitar um aumento de limite” em Service Quotas.

> **Note**
>
> Uma KPU adicional é cobrada para fins de orquestração. Para obter mais informações, consulte Preço do Managed Service for Apache Flink.

Atualizar o paralelismo do seu aplicativo

Esta seção contém exemplos de solicitações de ações de API que definem o paralelismo de um aplicativo. Para ver mais exemplos e instruções sobre como usar blocos de solicitação com ações de API, consulte Exemplo de código de API para o Managed Service for Apache Flink.

O exemplo a seguir de solicitação para a ação `CreateApplication` define o paralelismo quando você está criando um aplicativo:

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

O exemplo de solicitação a seguir para a ação `UpdateApplication` define o paralelismo para um aplicativo existente:

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

O exemplo de solicitação a seguir para a ação `UpdateApplication` desativa o paralelismo para um aplicativo existente:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
```

```
            "AutoScalingEnabledUpdate": "false"
         }
      }
   }
}
```

Escalabilidade automática

O Managed Service for Apache Flink dimensiona elasticamente o paralelismo de seu aplicativo para acomodar o throughput de dados de sua fonte e a complexidade de seu operador na maioria dos cenários. O Managed Service for Apache Flink monitora o uso de recursos (CPU) do seu aplicativo e aumenta ou diminui elasticamente o paralelismo do seu aplicativo de acordo com:

- Seu aplicativo aumenta (aumenta o paralelismo) se a CloudWatch métrica `containerCPUUtilization` for maior que 75 por cento ou mais por 15 minutos. Isso significa que a ação `ScaleUp` é acionada quando há 15 pontos de dados consecutivos com um período de 1 minuto igual ou superior a 75%.
- Seu aplicativo reduz a escala verticalmente (diminui o paralelismo) quando o uso da CPU permanece abaixo de 10% por seis horas. Isso significa que a ação `ScaleDown` é acionada quando há 360 pontos de dados consecutivos com um período de 1 minuto inferior a 10%.

> **Note**
>
> Um período máximo de `containerCPUUtilization` acima de 1 minuto pode ser referenciado para encontrar a correlação com um ponto de dados usado para a ação de escalonamento, mas não é necessário refletir o momento exato em que a ação é acionada.

O Managed Service for Apache Flink não reduzirá o valor `CurrentParallelism` do seu aplicativo para menos do que a configuração `Parallelism` do seu aplicativo.

Quando o serviço do Managed Service for Apache Flink estiver escalando seu aplicativo, ele estará no status `AUTOSCALING`. Você pode verificar o status atual da sua inscrição usando as ListApplicationsações DescribeApplicationou. Enquanto o serviço está escalando seu aplicativo, a única ação de API válida que você pode usar é StopApplicationcom o `Force` parâmetro definido como. `true`

Você pode usar a propriedade `AutoScalingEnabled` (parte de `FlinkApplicationConfiguration`) para ativar ou desativar o comportamento de ajuste de escala automático. Sua conta AWS é cobrada pelas KPUs que o Managed Service for Apache Flink provisiona, o que é uma função das configurações `parallelism` e `parallelismPerKPU` do seu aplicativo. Um pico de atividade aumenta os custos do Managed Service for Apache Flink.

Para obter mais informações sobre preços, consulte Preço do Amazon Managed Service for Apache Flink.

Observe o seguinte sobre escalonamento de aplicativo:

- O ajuste de escala automático está habilitado por padrão.
- O escalonamento não se aplica aos blocos de anotações do Studio. No entanto, se você implantar um bloco de anotações do Studio como um aplicativo de estado durável, o escalonamento será aplicado ao aplicativo implantado.
- Seu aplicativo tem um limite padrão de 64 KPUs. Para ter mais informações, consulte Quota.
- Quando o automáticoajuste de escala automático atualiza o paralelismo do aplicativo, o aplicativo passa por um tempo de inatividade. Para evitar esse tempo de inatividade, faça o seguinte:
  - Desabilitar o ajuste de escala automático
  - Configure seu aplicativo `parallelism` e `parallelismPerKPU` com a UpdateApplicationação. Para obter mais informações sobre como definir as configurações de paralelismo do aplicativo, consulte the section called “Atualizar o paralelismo do seu aplicativo” a seguir.
  - Monitore periodicamente o uso de recursos do seu aplicativo para verificar se ele tem as configurações de paralelismo corretas para seu workload. Para obter informações sobre monitoramento de alocação de recursos, consulte the section called “Métricas e dimensões no Managed Service for Apache Flink”.

Considerações sobre o MaxParallelism

- A lógica de autoescala evitará a escalabilidade de uma tarefa do Flink para um paralelismo que causará interferência na tarefa e no operador `maxParallelism`. Por exemplo, se for um trabalho simples com apenas uma origem e um coletor em que a origem tenha `maxParallelism` 16 e `sink` tenha 8, não escalaremos automaticamente o trabalho para acima de 8.
- Se `maxParallelism` não estiver definido para um trabalho, o padrão do Flink será 128. Portanto, se acha que um trabalho precisará ser executado em um paralelismo maior do que 128, você precisará definir esse número para seu aplicativo.

- Se você espera ver seu trabalho escalar automaticamente, mas não está vendo, certifique-se de que seus valores `maxParallelism` o permitam.

Para obter informações adicionais, consulte Monitoramento aprimorado e escalabilidade automática para o Apache Flink

Para obter um exemplo, consulte kda-flink-app-autoscaling.

Uso de tags

Esta seção descreve como adicionar tags de metadados de valor-chave para aplicativos do Managed Service for Apache Flink. Essas tags podem ser usadas para as seguintes finalidades:

- Determinar o faturamento para um aplicativo individual do Managed Service for Apache Flink. Para obter mais informações, consulte Como usar tags de alocação no Guia de Gerenciamento de custos e faturamento.
- Controlar o acesso a recursos de aplicativos com base em tags. Para obter mais informações, consulte Controlar o acesso usando tags no Guia do usuário do AWS Identity and Access Management.
- Finalidades definidas pelo usuário. Você pode definir a funcionalidade do aplicativo com base na presença de tags do usuário.

Observe as seguintes informações sobre a marcação:

- O número máximo de tags do aplicativo inclui tags de sistema. O número máximo de tags do aplicativo definidas pelo usuário é de 50.
- Se uma ação inclui uma lista de tags que tem valores `Key` duplicados, o serviço lançará um `InvalidArgumentException`.

Este tópico contém as seguintes seções:

Adicionar tags quando um aplicativo é criado

Você pode adicionar tags ao criar um aplicativo usando o parâmetro `tags` da ação CreateApplication.

O exemplo de solicitação a seguir mostra o nó `Tags` para uma solicitação `CreateApplication`:

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

Adicionar ou atualizar tags para um aplicativo existente

Você pode adicionar tags a um aplicativo usando a ação TagResource. Você não pode adicionar tags a um aplicativo usando a ação UpdateApplication.

Para atualizar uma tag existente, adicione uma tag com a mesma chave da tag existente.

O exemplo de solicitação a seguir para a ação `TagResource` adiciona novas tags ou atualiza tags existentes:

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

Listar tags para um aplicativo

Para listar tags existentes, use a ação ListTagsForResource.

O exemplo de solicitação a seguir para a ação `ListTagsForResource` lista as tags de um aplicativo:

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

Remover tags de um aplicativo

Para remover tags de um aplicativo, use a ação UntagResource.

O exemplo de solicitação a seguir para a ação `UntagResource` remove tags de um aplicativo:

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

Usar CloudFormation for Managed Service for Apache Flink

Os exercícios a seguir mostram como iniciar um aplicativo Flink criado através de AWS CloudFormation usando uma função do Lambda na mesma pilha.

Antes de começar

Antes de começar este exercício, siga as etapas de criação de um aplicativo Flink usando AWS CloudFormation em AWS::KinesisAnalytics::Application.

Redação de sua função do Lambda

Para iniciar um aplicativo Flink após a criação ou atualização, usamos a API kinesisanalyticsv2 start-application. A chamada será acionada por um evento AWS CloudFormation após a criação do aplicativo Flink. Discutiremos como configurar a pilha para acionar a função do Lambda

posteriormente neste exercício, mas primeiro nos concentraremos na declaração da função do Lambda e em seu código. Usamos o runtime `Python3.8` neste exemplo.

```
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)})
```

No código anterior, o Lambda processará os eventos AWS CloudFormation recebidos, filtrará tudo além de `Create` e `Update`, obterá o estado do aplicativo e o iniciará se o estado for `READY`. Para obter o estado do aplicativo, você precisa criar a função do Lambda, conforme mostrado a seguir:

Criação de uma função do Lambda

Você cria uma função para que o Lambda “converse” com sucesso com o aplicativo e grave logs. Essa função usará políticas gerenciadas padrão, mas talvez você queira restringi-la 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: /
```

Observe que os recursos do Lambda serão criados após a criação do aplicativo Flink na mesma pilha porque dependem dele.

Invocação da função do Lambda

Agora, tudo o que resta é invocar a função do Lambda. Isso é feito usando um 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
```

Isso é tudo o que você precisa para iniciar seu aplicativo Flink usando o Lambda. Agora você está pronto para criar sua própria pilha ou usar o exemplo completo abaixo para ver como todas essas etapas funcionam na prática.

Exemplo completo

O exemplo a seguir é uma versão ligeiramente estendida das etapas acima com um ajuste `RunConfiguration` adicional feito por meio dos parâmetros do modelo. Esta é uma pilha funcional para você experimentar. Não deixe de ler as notas anexas:

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:
                - kinesisanlaytics.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
```

Novamente, talvez você queira ajustar as funções do Lambda e do próprio aplicativo.

Antes de criar a pilha acima, não se esqueça de especificar seus parâmetros.

parâmetros.json

```
[
```

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

Substitua YOUR_BUCKET_ARN e YOUR_JAR pelos seus requisitos específicos. Você pode seguir este guia para criar um bucket do Amazon S3 e um jar de aplicativos.

Agora crie a pilha (substitua YOUR_REGION por uma região de sua escolha, por exemplo, 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
```

Agora você pode navegar até https://console.aws.amazon.com/cloudformation e ver o progresso. Depois de criado, você deverá ver seu aplicativo Flink no estado `Starting`. Pode demorar alguns minutos até que ele comece a `Running`.

Para obter mais informações, consulte as informações a seguir.

- Quatro maneiras de recuperar qualquer propriedade de serviço AWS usando o AWS CloudFormation (Parte 1 de 3).
- Demonstração: Pesquisar IDs de imagem de máquina da Amazon.

Usar o painel do Apache Flink com o Managed Service for Apache Flink

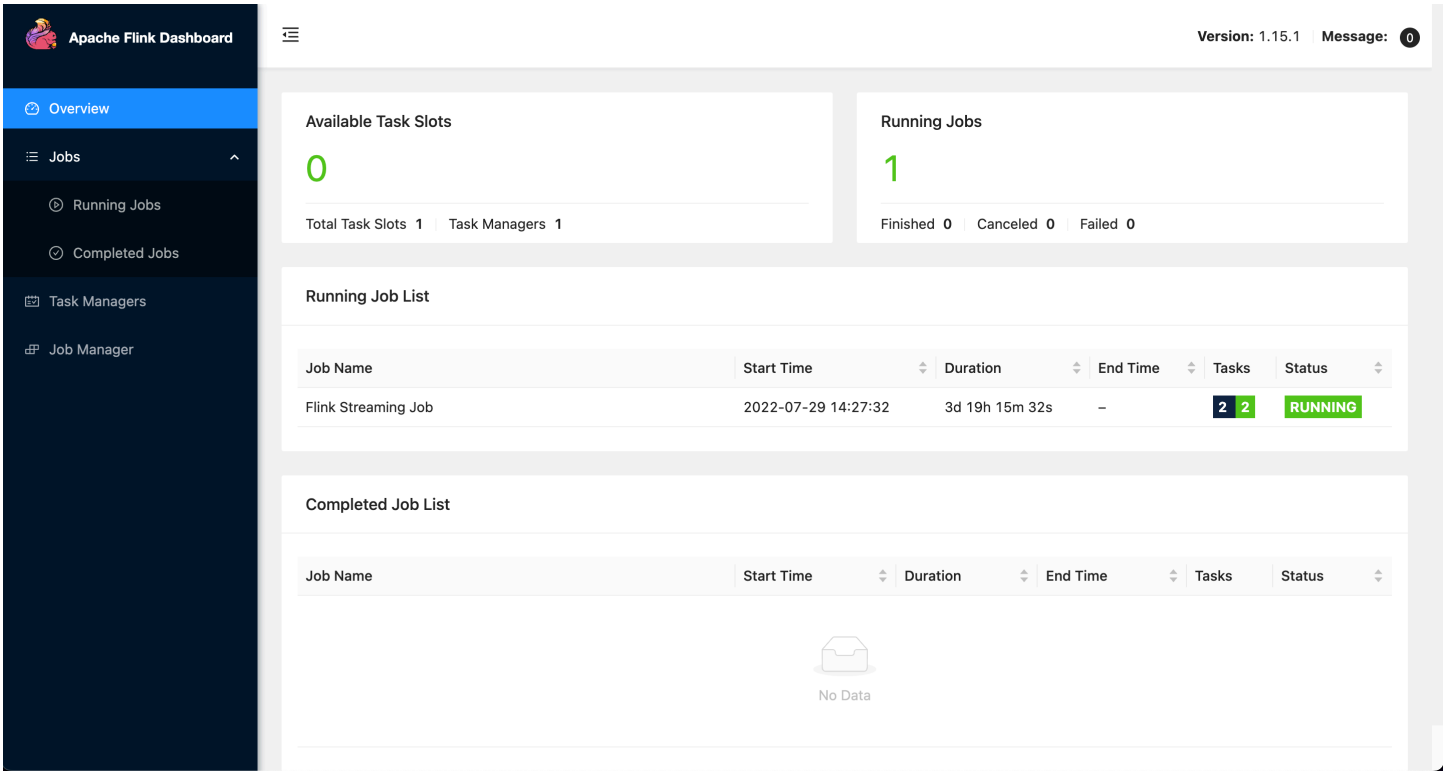
Você pode usar o painel do Apache Flink do seu aplicativo para monitorar seu serviço gerenciado quanto à integridade do aplicativo Apache Flink. O painel do seu aplicativo mostra as seguintes informações:

- Recursos em uso, incluindo gerenciadores de tarefas e slots de tarefas.
- Informações sobre trabalhos, incluindo aqueles que estão em execução, concluídos, cancelados e com falha.

Para obter informações sobre gerenciadores de tarefas, slots de tarefas e trabalhos do Apache Flink, consulte Arquitetura do Apache Flink no site do Apache Flink.

Observe o seguinte sobre o uso do painel do Apache Flink com os aplicativos do Managed Service para Apache Flink:

- O painel do Apache Flink para aplicativos do Managed Service for Apache Flink é somente para leitura. Você não pode alterar o seu aplicativo do Managed Service for Apache Flink usando o painel do Apache Flink.
- O painel do Apache Flink não é compatível com o Microsoft Internet Explorer.



Acessar o painel do Apache Flink do seu aplicativo

Você pode acessar o painel do Apache Flink do seu aplicativo por meio do console do Managed Service for Apache Flink ou solicitando um endpoint de URL seguro usando a CLI.

Acessar o painel do Apache Flink do seu aplicativo usando o console do Managed Service for Apache Flink

Para acessar o painel do Apache Flink do seu aplicativo a partir do console, escolha Painel do Apache Flink na página do seu aplicativo.

> **Note**
>
> Quando você abre o painel do console do Managed Service for Apache Flink, a URL gerada pelo console será válida por 12 horas.

Acessar o painel do Apache Flink do seu aplicativo usando a CLI do Managed Service for Apache Flink

Você pode usar a CLI do Managed Service for Apache Flink para gerar uma URL para acessar o painel do seu aplicativo. A URL que você gera é válida por um período especificado.

> **Note**
>
> Se você não acessar a URL gerada em três minutos, ela não será mais válida.

Você gera a URL do painel usando a ação CreateApplicationPresignedURL. Você especifica os seguintes parâmetros para a ação:

- O nome do aplicativo
- O tempo em segundos em que a URL será válida
- Você especifica `FLINK_DASHBOARD_URL` como o tipo de URL.

Versões de liberação

Este tópico contém informações sobre os recursos suportados e as versões de componentes recomendadas para cada versão do Managed Service for Apache Flink.

Amazon Managed Service for Apache Flink liberação 1.15.2

O Managed Service for Apache Flink suporta os seguintes novos recursos no Apache 1.15.2

Recurso	Descrição	Referência ao Apache FLIP
Coletor assíncrono	Uma estrutura AWS contribuída para criar destinos assíncronos que permite aos desenvolvedores criar conectores AWS personalizados com menos da metade do esforço anterior. Para obter mais informações, consulte The Generic Asynchronous Base Sink.	FLIP-171: Coletor assíncrono.
Coletor do Kinesis Data Firehose	AWS contribuiu com um novo coletor Amazon Kinesis Firehose usando a estrutura assíncrona.	Coletor Amazon Kinesis Data Firehose.
Interromper com o Savepoint	Interromper com Savepoint garante uma operação de parada limpa e, mais importante, oferecendo suporte a semântica de exatamente uma vez para clientes que confiam nela.	FLIP-34: Encerrar/suspender o trabalho com o Savepoint.

Recurso	Descrição	Referência ao Apache FLIP
Desacoplamento do Scala	Agora, os usuários podem aproveitar a API Java de qualquer versão do Scala, incluindo o Scala 3. Os clientes precisarão incluir a biblioteca padrão Scala de sua seleção em seus aplicativos Scala.	FLIP-28: Objetivo de longo prazo de tornar o Flink Table livre do Scala.
Scala	Veja desacoplamento do Scala acima	FLIP-28: Objetivo de longo prazo de tornar o Flink Table livre do Scala.
Métricas unificadas de conectores	O Flink definiu métricas padrão para trabalhos, tarefas e operadores. O Managed Service for Apache Flink continuará a oferecer suporte às métricas de coletor e de fonte e, na versão 1.15, será introduzido `numRestarts` paralelamente às métricas `fullRestarts` de disponibilidade.	FLIP-33: Padronizar as métricas do conector e FLIP-179: Expor métricas padronizadas do operador.
Verificação de tarefas concluídas	Esse recurso é ativado por padrão no Flink 1.15 e possibilita continuar executando pontos de verificação mesmo que partes do gráfico de trabalho tenham concluído o processamento de todos os dados, o que pode acontecer se ele contiver fontes limitadas (em lote).	FLIP-147: Pontos de verificação de suporte após a conclusão das tarefas.

Alterações no Amazon Managed Service for Apache Flink com o Apache Flink 1.15

Notebooks Studio

O Managed Service for Apache Flink Studio agora é compatível com o Apache Flink 1.15. O Managed Service for Apache Flink Studio utiliza blocos de anotações do Apache Zeppelin para fornecer uma experiência de desenvolvimento de interface única para desenvolvimento, depuração de código e execução de aplicativos de processamento de stream do Apache Flink. Você pode aprender mais sobre o Managed Service for Apache Flink Studio e os conceitos básicos em. Usar um notebook Studio com Managed Service for Apache Flink

Connector EFO

Ao atualizar para o Managed Service for Apache Flink versão 1.15, verifique se você está usando o conector EFO mais recente, ou seja, qualquer versão 1.15.3 ou mais recente. Para obter mais informações sobre o motivo, consulte FLINK-29324.

Desacoplamento do Scala

A partir do Flink 1.15.2, você precisará agrupar a biblioteca padrão Scala de sua seleção em seus aplicativos Scala.

Coletor Kinesis Data Firehose

Ao atualizar para o Managed Service for Apache Flink versão 1.15, certifique-se de usar o coletor Amazon Kinesis Data Firehose mais recente.

Conetores Kafka

Ao atualizar para o Amazon Managed Service for Apache Flink para o Apache Flink versão 1.15, verifique se você está usando as APIs mais recentes do conetor Kafka. O Apache Flink descontinuou o FlinkKafkaConsumer e o FlinkKafkaProducer. Essas APIs para o coletor Kafka não conseguem se restabelecer com o Kafka para o Flink 1.15. Verifique se você está usando o KafkaSource e o KafkaSink.

Componentes

Componente	Versão
Java	11 (recomendado)
Scala	2.12
Managed Service for Apache Flink Runtime (aws-kinesisanalytics-runtime)	1.2.0
AWS Conector Kinesis (flink-connector-kinesis)	1.15.4
Apache Beam (somente aplicativos Beam)	2.33.0, com Jackson versão 2.12.2

Usar um notebook Studio com Managed Service for Apache Flink

Os notebooks Studio para Managed Service for Apache Flink permitem que você consulte interativamente fluxos de dados em tempo real e crie e execute facilmente aplicativos de processamento de fluxo usando SQL, Python e Scala padrão. Com alguns cliques no AWS console de gerenciamento, você pode iniciar um notebook com tecnologia sem servidor para consultar fluxos de dados e obter resultados em segundos.

Um notebook é um ambiente de desenvolvimento baseado na web. Com os notebooks, você obtém uma experiência simples de desenvolvimento interativo combinada com os recursos avançados fornecidos pelo Apache Flink. Os notebooks Studio usam notebooks equipados com Apache Zeppelin e usam o Apache Flink como mecanismo de processamento de streaming. Os notebooks Studio combinam perfeitamente essas tecnologias para tornar a análise avançada em fluxos de dados acessível a desenvolvedores de todos os conjuntos de habilidades.

O Apache Zeppelin fornece aos seus notebooks Studio um conjunto completo de ferramentas de análise, incluindo as seguintes:

- Visualização de dados
- Exportar dados para arquivos
- Controlar o formato da saída para facilitar a análise

Para começar a usar o Managed Service for Apache Flink e Apache Zeppelin, consulte. Tutorial de criação de um bloco de anotações Studio Para obter mais informações sobre o Apache Zeppelin, consulte a documentação do Apache Zeppelin.

Com um notebook, você modela consultas usando a API e SQL do Apache Flink Table em SQL, Python ou Scala, ou API em Scala. DataStream Com alguns cliques, você pode promover em seguida o notebook Studio a um aplicativo de processamento de fluxo do Managed Service for Apache Flink, em execução contínua, para seus workloads de produção.

Este tópico contém as seguintes seções:

- Criar um notebook Studio
- Análise interativa de dados de streaming

Criar um notebook Studio

Um notebook Studio contém consultas ou programas escritos em SQL, Python ou Scala que são executados em dados de streaming e retornam resultados analíticos. Você cria seu aplicativo usando o console ou a CLI e fornece consultas para analisar os dados da sua fonte de dados.

Seu aplicativo tem os seguintes componentes:

- Uma fonte de dados, como um cluster do Amazon MSK, um fluxo de dados do Kinesis ou um bucket do Amazon S3.
- Um banco de dados do AWS Glue. Esse banco de dados contém tabelas, que armazenam esquemas e endpoints de origem e destino de seus dados. Para ter mais informações, consulte Trabalhar com o AWS Glue.
- Código do seu aplicativo Seu código implementa sua consulta ou programa de análise.
- Suas configurações do aplicativo e propriedades de runtime. Para obter informações sobre configurações do aplicativo e propriedades de runtime, consulte os seguintes tópicos no Guia do desenvolvedor para aplicativos Apache Flink:
  - Paralelismo e escalonamento de aplicativos: você usa a configuração de paralelismo do seu aplicativo para controlar o número de consultas que seu aplicativo pode executar simultaneamente. Suas consultas também podem aproveitar o aumento do paralelismo se tiverem vários caminhos de execução, como nas seguintes circunstâncias:
    - Ao processar vários fragmentos de um fluxo de dados do Kinesis
    - Ao particionar dados usando o operador `KeyBy`.

    - Ao usar vários operadores de janela

    Para obter mais informações sobre o escalonamento de aplicativos, consulte Escalonamento de aplicativos no Managed Service for Apache Flink.
- Logs e monitoramento: para obter informações sobre logs e monitoramento de aplicativos, consulte Logs e monitoramento no Amazon Managed Service for Apache Flink para Apache Flink.
- Seu aplicativo usa pontos de verificação e pontos de salvamento para tolerância a falhas. Os pontos de verificação e os pontos de salvamento não estão habilitados por padrão para notebooks Studio.

Você pode criar seu notebook Studio usando o AWS Management Console ou o AWS CLI.

Ao criar o aplicativo a partir do console, você tem as seguintes opções:

- No console do Amazon MSK, selecione seu cluster e, em seguida, selecione Processar dados em tempo real.
- No console do Kinesis Data Streams, selecione seu fluxo de dados e, em seguida, na aba Aplicativos, selecione Processar dados em tempo real.
- No console Managed Service for Apache Flink, selecione a aba Studio e, em seguida, selecione Criar notebook Studio.

Para ver um tutorial, consulte Detecção de eventos com serviço gerenciado para Apache Flink.

Para ver um exemplo de uma solução de notebook Studio mais avançada, consulte Apache Flink no Amazon Managed Service for Apache Flink Studio.

Análise interativa de dados de streaming

Você usa um notebook com tecnologia sem servidor e com tecnologia Apache Zeppelin para interagir com seus dados de streaming. Seu notebook pode ter várias notas, e cada nota pode ter um ou mais parágrafos onde você pode escrever seu código.

O exemplo de consulta SQL a seguir mostra como recuperar dados de uma fonte de dados:

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

```
select * from stock;
```

Para obter mais exemplos de consultas SQL do Flink Streaming, consulte Exemplos e tutoriaise Consultas na Documentação do Apache Flink.

Você pode usar as consultas SQL do Flink no notebook Studio para consultar dados de streaming. Você também pode usar Python (API de tabela) e Scala (APIs de tabela e fluxo de dados) para escrever programas para consultar seus dados de streaming de forma interativa. Você pode visualizar os resultados de suas consultas ou programas, atualizá-los em segundos e executá-los novamente para ver os resultados atualizados.

Intérpretes Flink

Você especifica qual linguagem o Managed Service for Apache Flink usa para executar seu aplicativo usando um intérprete. Você pode usar os seguintes intérpretes com o Managed Service for Apache Flink:

Nome	Classe	Descrição
%flink	FlinkInterpreter	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment
%flink.pyflink	PyFlinkInterpreter	Provides a python environment
%flink.ipyflink	IPyFlinkInterpreter	Provides an ipython environment
%flink.ssql	FlinkStreamSqlInterpreter	Provides a stream sql environment
%flink.bsql	FlinkBatchSqlInterpreter	Provides a batch sql environment

Para obter mais informações sobre intérpretes Flink, consulte Interpretador Flink for Apache Zeppelin.

Se você estiver usando `%flink.pyflink` ou `%flink.ipyflink` como intérpretes, precisará usar o `ZeppelinContext` para visualizar os resultados no caderno.

Para exemplos mais PyFlink específicos, consulte Consulte seus fluxos de dados de forma interativa usando o Managed Service para Apache Flink Studio e Python.

Variáveis de ambiente da tabela Apache Flink

O Apache Zeppelin fornece acesso aos recursos do ambiente de tabela usando variáveis de ambiente.

Você acessa os recursos do ambiente de tabela Scala com as seguintes variáveis:

Variável	Recurso
`senv`	`StreamExecutionEnvironment`
`stenv`	`StreamTableEnvironment para blink planner`

Você acessa os recursos do ambiente de tabela Python com as seguintes variáveis:

Variável	Recurso
`s_env`	`StreamExecutionEnvironment`
`st_env`	`StreamTableEnvironment para blink planner`

Para obter mais informações sobre o uso de ambientes de tabela, consulte Create a TableEnvironment na documentação do Apache Flink.

Implantação como um aplicativo com estado durável

Você pode criar seu código e exportá-lo para o Amazon S3. Você pode promover o código que escreveu em sua nota para um aplicativo de processamento de streams em execução contínua. Há dois modos de executar um aplicativo Apache Flink no Managed Service for Apache Flink: com um notebook Studio, você tem a capacidade de desenvolver seu código de forma interativa, visualizar

os resultados do seu código em tempo real e visualizá-lo em sua nota. Depois de implantar uma nota para execução no modo de streaming, o Managed Service for Apache Flink cria um aplicativo para você que é executado continuamente, lê dados de suas fontes, grava em seus destinos, mantém o estado do aplicativo de longa duração e escala automaticamente com base no throughput de seus fluxos de origem.

> **Note**
>
> O bucket do S3 para o qual você exporta o código do aplicativo deve estar na mesma região que o notebook Studio.

Você só pode implantar uma nota do seu notebook Studio se ela atender aos seguintes critérios:

- Os parágrafos devem ser ordenados sequencialmente. Quando você implanta seu aplicativo, todos os parágrafos em uma nota serão executados sequencialmente (left-to-right, top-to-bottom) conforme aparecem na sua nota. Você pode verificar essa ordem escolhendo Executar todos os parágrafos em sua nota.
- Seu código é uma combinação de Python e SQL ou Scala e SQL. No momento, não oferecemos suporte para Python e Scala juntos. deploy-as-application
- Sua nota deve ter apenas os seguintes intérpretes:`%flink`, `%flink.ssql`, `%flink.pyflink`, `%flink.ipyflink`, `%md`.
- O uso do objeto de contexto do Zeppelin não z é suportado. Métodos que não retornam nada farão nada, exceto registrar um aviso. Outros métodos gerarão exceções do Python ou falharão na compilação em Scala.
- Uma nota deve resultar em uma única tarefa do Apache Flink.
- Não há suporte para a implantação de notas com formulários dinâmicos como um aplicativo.
- Os parágrafos %md (Markdown) serão ignorados na implantação como um aplicativo, pois espera-se que contenham documentação legível por humanos que não é adequada para execução como parte do aplicativo resultante.
- Os parágrafos desativados para execução no Zeppelin serão ignorados na implantação como um aplicativo. Mesmo que um parágrafo desativado use um intérprete incompatível, por exemplo, `%flink.ipyflink` em uma nota com `%flink and %flink.ssql` intérpretes, ele será ignorado durante a implantação da nota como um aplicativo e não resultará em erro.
- Deve haver pelo menos um parágrafo presente com o código-fonte (Flink SQL PyFlink ou Flink Scala) habilitado para execução para que a implantação do aplicativo seja bem-sucedida.

- Definir o paralelismo na diretiva do intérprete dentro de um parágrafo (por exemplo`%flink.ssql(parallelism=32)`) será ignorado em aplicativos implantados a partir de uma nota. Em vez disso, você pode atualizar o aplicativo implantado por meio daAWS Management Console, AWS Command Line Interface ou AWS API para alterar as configurações de paralelismo e/ou ParallelismPer KPU de acordo com o nível de paralelismo que seu aplicativo exige, ou você pode ativar o escalonamento automático para seu aplicativo implantado.
- Se você estiver implantando como um aplicativo com estado durável, sua VPC deverá ter acesso à Internet. Se sua VPC não tiver acesso à Internet, consulte. Implantar como um aplicativo com estado durável em uma VPC sem acesso à Internet

Critérios Scala/Python

- Em seu código Scala ou Python, use o planejador Blink (`senv`, `stenv` para Scalas_env; `st_env` para Python) e não o planejador "Flink" antigo (`stenv_2` para Scala, `st_env_2` para Python). O projeto Apache Flink recomenda o uso do planejador Blink para casos de uso de produção, e esse é o planejador padrão no Zeppelin e no Flink.
- Seus parágrafos do Python não devem usar invocações/atribuições de shell usando `!` ou comandos mágicos do IPython como `%timeit` ou `%conda` em notas destinadas a serem implantadas como aplicativos.
- Você não pode usar classes de casos do Scala como parâmetros de funções passadas para operadores de fluxo de dados de ordem superior, como `map` e `filter`. Para obter informações sobre as classes de casos do Scala, consulte CLASSES DE CASO na documentação do Scala.

Critérios SQL

- Declarações SELECT simples não são permitidas, pois não há nada equivalente à seção de saída de um parágrafo em que os dados possam ser entregues.
- Em qualquer parágrafo, as declarações DDL (`USE`, `CREATE`, `ALTER` `DROP`, `SET`, `RESET`) devem preceder as declarações DML (`INSERT`). Isso ocorre porque as declarações DML em um parágrafo devem ser enviadas juntas como uma única tarefa do Flink.
- Deve haver no máximo um parágrafo que contenha declarações DML. Isso porque, para o deploy-as-application recurso, oferecemos suporte apenas ao envio de um único trabalho para o Flink.

Para obter mais informações e um exemplo, consulte Traduzir, redigir e analisar dados de streaming usando funções SQL com o Amazon Managed Service for Apache Flink, Amazon Translate e Amazon Comprehend.

Permissões do IAM para notebooks Studio

O Managed Service for Apache Flink cria uma função do IAM para você quando você cria um notebook Studio por meio do AWS Management Console. Ele também associa a essa função uma política que permite o seguinte acesso:

Serviço	Acesso
CloudWatch Registros	Lista
Amazon EC2	Lista
AWS Glue	Ler, Gravar
Managed Service for Apache Flink	Ler
Managed Service for Apache Flink V2	Leitura
Amazon S3	Ler, Gravar

Conectores e dependências

Os conectores permitem que você leia e grave dados em várias tecnologias. O Managed Service for Apache Flink agrupa três conectores padrão com seu notebook Studio. Também é possível usar conectores personalizados. Para obter mais informações sobre coletores de tabelas, consulte Tabela e conectores SQL na Documentação do Apache Flink.

Conectores padrão

Se você usar o AWS Management Console para criar seu notebook Studio, o Managed Service for Apache Flink inclui os seguintes conectores personalizados por padrão: `flink-sql-connector-flink`, `flink-connector-kafka_2.12` e `aws-msk-iam-auth`. Para criar um notebook Studio por meio do console sem esses conectores personalizados, escolha a opção Criar com

configurações personalizadas. Em seguida, ao acessar a página Configurações, desmarque as caixas de seleção ao lado dos dois conectores.

Se você usa a CreateApplicationAPI para criar seu notebook Studio, os `flink-connector-kafka` conectores `flink-sql-connector-flink` e não são incluídos por padrão. Para adicioná-los, especifique-os como a `MavenRefernce` no tipo de dados `CustomArtifactsConfiguration`, conforme mostrado nos exemplos a seguir.

O conector `aws-msk-iam-auth` é o conector a ser usado com o Amazon MSK, que inclui o recurso de autenticação automática com o IAM.

> **Note**
>
> As versões do conector mostradas no exemplo a seguir são as únicas para as quais oferecemos suporte.

```
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 adicionar esses conectores a um notebook existente, use a operação da UpdateApplicationAPI e especifique-os como a `MavenReference` no tipo de `CustomArtifactsConfigurationUpdate` dados.

> **Note**
>
> Você pode definir como verdadeiro `failOnError` para o `flink-sql-connector-kinesis` conector na API da tabela.

Dependências e conectores personalizados

Para usar o AWS Management Console para adicionar uma dependência ou um conector personalizado ao seu notebook Studio, siga estas etapas:

1. Carregue seu arquivo do conector personalizado no Amazon S3.
2. No AWS Management Console, selecione a opção Criação personalizada para criar seu notebook Studio.
3. Siga o fluxo de trabalho de criação do notebook Studio até chegar à etapa Configurações.
4. Na seção Conectores personalizados, selecione Adicionar conector personalizado.
5. Especifique a localização da dependência ou conector personalizado no Amazon S3.
6. Escolha Salvar alterações.

Para adicionar um JAR de dependência ou um conector personalizado ao criar um novo notebook Studio usando a CreateApplicationAPI, especifique a localização do JAR de dependência no Amazon S3 ou o conector personalizado no `CustomArtifactsConfiguration` tipo de dados. Para

adicionar uma dependência ou um conector personalizado a um notebook Studio existente, invoque a operação de UpdateApplicationAPI e especifique a localização do JAR da dependência no Amazon S3 ou o conector personalizado no tipo de dados. `CustomArtifactsConfigurationUpdate`

> **Note**
>
> Ao incluir uma dependência ou um conector personalizado, você também deve incluir todas as dependências transitivas que não estão agrupadas nela.

Funções definidas pelo usuário

As funções definidas pelo usuário (UDFs) são pontos de extensão que permitem chamar a lógica usada com frequência ou a lógica personalizada que não pode ser expressa de outra forma em consultas. Você pode usar Python ou uma linguagem JVM como Java ou Scala para implementar suas UDFs em parágrafos dentro do seu notebook Studio. Você também pode adicionar ao seu notebook Studio arquivos JAR externos que contêm UDFs implementadas em uma linguagem JVM.

Ao implementar JARs que registram classes abstratas dessa subclasse `UserDefinedFunction` (ou suas próprias classes abstratas), use o escopo fornecido no Apache Maven, as declarações de dependência `compileOnly` no Gradle, o escopo fornecido no SBT ou uma diretiva equivalente na configuração de compilação do projeto UDF. Isso permite que o código-fonte da UDF seja compilado com as APIs do Flink, mas as classes da API do Flink não estão incluídas nos artefatos de construção. Consulte esse exemplo de pom do exemplo jar UDF, que atende a esse pré-requisito em um projeto Maven.

> **Note**
>
> Para obter mais informações e um exemplo, consulte Traduzir, redigir e analisar dados de streaming usando funções SQL com o Amazon Managed Service for Apache Flink, Amazon Translate e Amazon Comprehend no AWS Blog Machine Learning.

Para usar o console para adicionar arquivos JAR UDF ao seu notebook Studio, siga estas etapas:

1. Faça upload do arquivo JAR UDF para o Amazon S3.
2. No AWS Management Console, selecione a opção Criação personalizada para criar seu notebook Studio.

3. Siga o fluxo de trabalho de criação do notebook Studio até chegar à etapa Configurações.
4. Na seção Funções definidas pelo usuário, selecione Adicionar função definida pelo usuário.
5. Especifique a localização do Amazon S3 do arquivo JAR ou do arquivo ZIP que tem a implementação da sua UDF.
6. Escolha Salvar alterações.

Para adicionar um JAR UDF ao criar um novo notebook Studio usando a CreateApplicationAPI, especifique a localização do JAR no tipo de `CustomArtifactConfiguration` dados. Para adicionar um JAR UDF a um notebook Studio existente, invoque a operação da UpdateApplicationAPI e especifique a localização do JAR no tipo de `CustomArtifactsConfigurationUpdate` dados. Como alternativa, você pode usar o AWS Management Console para adicionar arquivos JAR UDF ao notebook do Studio.

Considerações com funções definidas pelo usuário

- O Managed Service for Apache Flink Studio usa a terminologia do Apache Zeppelin, em que um notebook é uma instância do Zeppelin que pode conter várias notas. Cada nota pode conter vários parágrafos. Com o Managed Service for Apache Flink Studio, o processo do intérprete é compartilhado em todas as notas no notebook. Portanto, se você realizar um registro de função explícito usando createTemporarySystemFunction em uma nota, a mesma poderá ser referenciada como está em outra nota do mesmo caderno.

  No entanto, a operação Implantar como aplicativo funciona em uma nota individual e não em todas as notas no notebook. Quando você executa a implantação como aplicativo, somente o conteúdo da nota ativa é usado para gerar o aplicativo. Qualquer registro explícito de função realizado em outros notebooks não faz parte das dependências de aplicativos geradas. Além disso, durante a opção Implantar como aplicativo, ocorre um registro implícito da função convertendo o nome da classe principal do JAR em uma string minúscula.

  Por exemplo, se `TextAnalyticsUDF` for a classe principal do UDF JAR, um registro implícito resultará no nome da função `textanalyticsudf`. Portanto, se um registro de função explícito na nota 1 do Studio ocorrer da seguinte forma, todas as outras notas nesse notebook (digamos, nota 2) poderão referenciar a função pelo nome por `myNewFuncNameForClass` causa do interpretador compartilhado:

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

No entanto, durante a operação de implantação como aplicativo na nota 2, esse registro explícito não será incluído nas dependências e, portanto, o aplicativo implantado não funcionará conforme o esperado. Por causa do registro implícito, por padrão, espera-se que todas as referências a essa função estejam com `textanalyticsudf` ou não `myNewFuncNameForClass`.

Se houver necessidade de registro de nome de função personalizado, espera-se que a própria nota 2 contenha outro parágrafo para realizar outro registro explícito da seguinte forma:

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

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

- Se o seu UDF JAR incluir SDKs do Flink, configure seu projeto Java para que o código-fonte do UDF possa ser compilado com os SDKs do Flink, mas as classes do SDK do Flink não sejam incluídas no artefato de construção, por exemplo, o JAR.

  Você pode usar o `provided` escopo no Apache Maven`compileOnly`, as declarações de dependência no Gradle, o escopo `provided` no SBT ou a diretiva equivalente na configuração de compilação do projeto UDF. Você pode consultar esse pom do exemplo UDF jar, que atende a esse pré-requisito em um projeto Maven. Para ver um step-by-step tutorial completo, consulte este artigo Traduza, edite e analise dados de streaming usando funções SQL com o Amazon Managed Service para Apache Flink, Amazon Translate e Amazon Comprehend.

Habilitar o checkpoint

Você ativa o ponto de verificação usando as configurações do ambiente. Para obter informações sobre pontos de verificação, consulte Tolerância a falhas no Guia do desenvolvedor do Managed Service for Apache Flink.

Definir o intervalo de verificação

O exemplo de código Scala a seguir define o intervalo do ponto de verificação do seu aplicativo em um minuto:

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

O exemplo de código Python a seguir define o intervalo do ponto de verificação do seu aplicativo em um minuto:

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

Definir o tipo de ponto de verificação

O exemplo de código Scala a seguir define o modo de ponto de verificação do seu aplicativo como `EXACTLY_ONCE` (o padrão):

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

O exemplo de código Python a seguir define o modo de ponto de verificação do seu aplicativo como `EXACTLY_ONCE` (o padrão):

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

Trabalhar com o AWS Glue

Seu notebook Studio armazena e obtém informações sobre suas fontes de dados e coletores do AWS Glue. Ao criar seu notebook Studio, você especifica o banco de dados AWS Glue que contém suas informações de conexão. Ao acessar suas fontes de dados e coletores, você especifica as tabelas AWS Glue contidas no banco de dados. Suas AWS Glue tabelas fornecem acesso às AWS Glue conexões que definem os locais, esquemas e parâmetros de suas fontes de dados e destinos.

Os notebooks Studio usam propriedades de tabela para armazenar dados específicos do aplicativo. Para ter mais informações, consulte Propriedades da tabela.

Para ver um exemplo de como configurar uma AWS Glue conexão, um banco de dados e uma tabela para uso com notebooks do Studio, consulte Criar um banco de dados da AWS Glue no tutorial Tutorial de criação de um bloco de anotações Studio.

Propriedades da tabela

Além dos campos de dados, suas tabelas AWS Glue fornecem outras informações ao seu notebook Studio usando as propriedades da tabela. O Managed Service for Apache Flink usa as seguintes propriedades da tabela AWS Glue:

- Usar valores de tempo do Apache Flink: essas propriedades definem como o Managed Service for Apache Flink emite valores de tempo de processamento de dados internos do Apache Flink.
- Usar o conector Flink e as propriedades de formato: essas propriedades fornecem informações sobre seus fluxos de dados.

Para adicionar uma propriedade a uma tabela AWS Glue, faça o seguinte:

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Na lista de tabelas, escolha a tabela que o aplicativo usa para armazenar suas informações de conexão de dados. Selecione Ação, Editar detalhes da tabela.
3. Em Propriedades da tabela, insira **`managed-flink.proctime`** para chave e **`user_action_time`** para Valor.

Usar valores de tempo do Apache Flink

O Apache Flink fornece valores de tempo que descrevem quando os eventos de processamento do stream ocorreram, como Tempo de Processamento e Tempo do Evento. Para incluir esses valores na saída do seu aplicativo, você define propriedades em sua AWS Glue tabela que instruem o runtime do Managed Service for Apache Flink a emitir esses valores nos campos especificados.

As chaves e os valores que você usa nas propriedades da tabela são os seguintes:

Tipo Timestamp	Chave	Valor
Tempo de processamento	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.
Hora do 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.*column_name*.milliseconds	The watermark interval in milliseconds

Usar o conector Flink e as propriedades de formato

Você fornece informações sobre suas fontes de dados aos conectores Flink do seu aplicativo usando as propriedades da tabela AWS Glue. Alguns exemplos das propriedades que o Managed Service for Apache Flink usa para conectores são os seguintes:

Tipo de conector	Chave	Valor
Kafka	`format`	The format used to deserialize and serialize Kafka messages, e.g. `json` or `csv`.
	`scan.startup.mode`	The startup mode for the Kafka consumer, e.g. `compensação mais antiga` or `timestamp` .
Kinesis	`format`	The format used to deserialize and serialize Kinesis data

Tipo de conector	Chave	Valor
		stream records, e.g. `json` or `csv`.
	`aws.region`	The AWS region where the stream is defined.
S3 (sistema de arquivos)	format	The format used to deserialize and serialize files, e.g. `json` or `csv`.
	`caminho`	The Amazon S3 path, e.g. `s3://mybucket/ .`

Para obter mais informações sobre outros conectores além do Kinesis e do Apache Kafka, consulte a documentação do seu conector.

Exemplos e tutoriais

Tópicos

- Tutorial: Criar um bloco de anotações Studio no Managed Service for Apache Flink
- Tutorial: implantação como um aplicativo com estado durável
- Exemplos

Tutorial: Criar um bloco de anotações Studio no Managed Service for Apache Flink

O tutorial a seguir demonstra como criar um bloco de anotações Studio que lê dados de um Kinesis Data Streams ou de um cluster Amazon MSK.

Este tutorial contém as seguintes seções:

- Configuração
- Criar um banco de dados da AWS Glue
- Próximas etapas

- Criar um bloco de anotações do Studio com o Kinesis Data Streams
- Criar um bloco de anotações do Studio com o Amazon MSK
- Limpar seu aplicativo e os recursos dependentes

Configuração

Verifique se a sua AWS CLI é a versão 2 ou posterior. Para instalar a AWS CLI mais recente, consulte Instalar, atualizar e desinstalar a AWS CLI versão 2.

Criar um banco de dados da AWS Glue

Seu bloco de anotações do Studio usa um banco de dados AWS Glue para metadados sobre sua fonte de dados Amazon MSK.

Criar um banco de dados da AWS Glue

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Selecione Add database (Adicionar banco de dados). Na janela Adicionar banco de dados, insira **`default`** para Nome do banco de dados. Selecione Create (Criar).

Próximas etapas

Com este tutorial, você pode criar um bloco de anotações Studio que usa o Kinesis Data Streams ou o Amazon MSK:

- Kinesis Data Streams: com o Kinesis Data Streams, você cria rapidamente um aplicativo que usa um fluxo de dados do Kinesis como uma fonte. Você só precisa criar um fluxo de dados do Kinesis como um recurso dependente.
- Amazon MSK: Com o Amazon MSK, você cria um aplicativo que usa um cluster do Amazon MSK como fonte. Você precisa criar uma Amazon VPC, uma instância cliente do Amazon EC2 e um cluster do Amazon MSK como recursos dependentes.

Criar um bloco de anotações do Studio com o Kinesis Data Streams

Este tutorial descreve como criar um bloco de anotações do Studio que usa um fluxo de dados do Kinesis como uma fonte.

Este tutorial contém as seguintes seções:

- Configuração
- Criar uma tabela do AWS Glue
- Criar um bloco de anotações do Studio com o Kinesis Data Streams
- Envie dados para o Kinesis Data Streams
- Teste seu bloco de anotações do Studio

Configuração

Antes de criar um bloco de anotações do Studio, crie um fluxo de dados do Kinesis (`ExampleInputStream`). Seu aplicativo usa esse fluxo para a fonte do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou este comando AWS CLI. Para obter instruções do console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie o fluxo **`ExampleInputStream`** e defina o Número de fragmentos abertos como **`1`**.

Para criar o fluxo (`ExampleInputStream`) usando o comando AWS CLI use o seguinte comando do Amazon Kinesis `create-stream` AWS CLI.

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

Criar uma tabela do AWS Glue

Seu bloco de anotações do Studio usa um AWS Gluebanco de dados para metadados sobre sua fonte de dados do Kinesis Data Streams.

> **Note**
>
> Você pode criar o banco de dados manualmente primeiro ou deixar que o Managed Service for Apache Flink o crie para você ao criar o bloco de anotações. Da mesma forma, você pode criar manualmente a tabela conforme descrito nesta seção ou usar o código do conector de criação de tabela para o Managed Service for Apache Flink em seu bloco de anotações no

Apache Zeppelin para criar sua tabela por meio de uma instrução DDL. Em seguida, você pode fazer o check-in no AWS Glue para garantir que a tabela foi criada corretamente.

Criar uma tabela

1. Faça logon no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Se você ainda não tiver um banco de dados AWS Glue, selecione Bancos de dados na barra de navegação à esquerda. Selecione Adicionar banco de dados. Na janela Adicionar banco de dados, insira **`default`** para Nome do banco de dados. Selecione Create (Criar).
3. Na barra de navegação à esquerda, selecione Tabelas. Na página Tabelas, selecione Adicionar tabelas, Adicionar tabela manualmente.
4. Na página Configurar propriedades da tabela, insira **`stock`** como nome da tabela. Certifique-se de selecionar o banco de dados que você criou anteriormente. Selecione Next (Próximo).
5. Na página Adicionar um armazenamento de dados, selecione Kinesis. Para Nome do fluxo, insira **`ExampleInputStream`**. Para URL de origem do Kinesis, selecione inserir **`https://kinesis.us-east-1.amazonaws.com`**. Se você copiar e colar o URL de origem do Kinesis, certifique-se de excluir todos os espaços à esquerda ou à direita. Selecione Next (Próximo).
6. Na página Classificação, selecione JSON. Selecione Next (Próximo).
7. Na página Definir um esquema, selecione Adicionar coluna para adicionar uma coluna. Adicione colunas com as seguintes propriedades:

Nome da coluna	Tipo de dados
`marcador de tempo`	**`string`**
`preço`	**`duplo`**

   Selecione Next (Próximo).
8. Na próxima página, verifique suas configurações e selecione Concluir.
9. Selecione a tabela recém-criada na lista de tabelas.
10. Selecione Editar tabela e adicione uma propriedade com a chave `managed-flink.proctime` e o valor `proctime`.

11. Selecione Apply (Aplicar).

Criar um bloco de anotações do Studio com o Kinesis Data Streams

Agora que você criou os recursos que o seu aplicativo usa, crie seu bloco de anotações do Studio.

Para criar seu aplicativo, você pode usar AWS Management Console ou AWS CLI.

- Crie um bloco de anotações do Studio usando AWS Management Console
- Crie um bloco de anotações do Studio usando AWS CLI

Crie um bloco de anotações do Studio usando AWS Management Console

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Na página dos Aplicativos do Managed Service for Apache Flink, selecione a guia Studio. Selecione Criar bloco de anotações do Studio.

   > **Note**
   >
   > Você também pode criar um bloco de anotações do Studio a partir dos consoles Amazon MSK ou Kinesis Data Streams, selecionando seu cluster Amazon MSK de entrada ou Kinesis Data Streams e selecionando Processar dados em tempo real.

3. Na página Criar bloco de anotações do Studio, forneça as seguintes informações:

   - Digite **`MyNotebook`** como nome do bloco de anotações.
   - Selecione padrão para o AWSbanco de dados do Glue.

   Selecione Criar bloco de anotações do Studio.
4. Na MyNotebookpágina, escolha Executar. Aguarde até que o Status mostre Em execução. As cobranças se aplicam quando o bloco de anotações está em execução.

Crie um bloco de anotações do Studio usando AWS CLI

Para criar seu bloco de anotações do Studio usando AWS CLI, faça o seguinte:

1. Verifique o seu ID da conta. Você precisa desse valor para criar seu aplicativo.

2. Crie a função `arn:aws:iam::AccountID:role/ZeppelinRole` e adicione as seguintes permissões à função criada automaticamente pelo console.

   ```
   "kinesis:GetShardIterator",

   "kinesis:GetRecords",

   "kinesis:ListShards"
   ```

3. Crie um arquivo chamado `create.json` com o conteúdo a seguir. Substitua os valores de espaço reservado por suas próprias informações.

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

4. Para criar o seu aplicativo, execute o comando a seguir:

   ```
   aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
   ```

5. Quando o comando for concluído, você verá uma saída que mostra os detalhes do seu novo bloco de anotações do Studio. A seguir, veja um exemplo de saída.

   ```
   {
       "ApplicationDetail": {
           "ApplicationARN": "arn:aws:kinesisanalyticsus-
   east-1:012345678901:application/MyNotebook",
   ```

```
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

6. Execute o comando a seguir para iniciar o aplicativo. Substitua o valor do exemplo pelo ID de sua conta.

```
aws kinesisanalyticsv2 start-application --application-arn
 arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

Envie dados para o Kinesis Data Streams

Para enviar dados de teste para seu Kinesis Data Streams, faça o seguinte:

1. Use o Kinesis Data Generator.
2. Escolha Criar um usuário Cognito com. CloudFormation
3. O console AWS CloudFormation é aberto com o modelo do Kinesis Data Generator. Selecione Next (Próximo).
4. Na página Especificar os detalhes da pilha, insira o seu nome de usuário e a senha para seu usuário do Cognito. Selecione Next (Próximo).
5. Na página Configurar opções de pilha, selecione Próximo.
6. Na página Review Kinesis-Data-Generator-Cognito-User, escolha a opção Eu reconheço que pode criar recursos do IAM. AWS CloudFormation caixa de seleção. Selecione Create Stack (Criar pilha).
7. Aguarde até que a criação da pilha do AWS CloudFormation seja concluída. Depois que a pilha estiver concluída, abra a pilha Usuário Cognito do Kinesis Data Generator no console AWS CloudFormation e selecione a guia Saídas. Abra o URL listado para o valor KinesisDataGeneratorUrlde saída.
8. Na página do Amazon Kinesis Data Generator, faça login com as credenciais que você criou na etapa 4.
9. Na página seguinte, forneça os valores a seguir:

Região	**us-east-1**

Fluxo do Amazon Kinesis Data Firehose	**ExampleInputStream**
Registros por segundo	**1**

Para Modelo de registro, cole o seguinte código:

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

10. Selecione Enviar dados.
11. O gerador enviará dados para o seu Kinesis Data Streams.

    Deixe o gerador executando enquanto você conclui a próxima seção.

Teste seu bloco de anotações do Studio

Nesta seção, você usa seu bloco de anotações do Studio para consultar dados do seu Kinesis Data Streams.

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Na página Aplicativos do Managed Service for Apache Flink, selecione a guia bloco de anotações do Studio. Escolha MyNotebook.
3. Na MyNotebookpágina, escolha Abrir no Apache Zeppelin.

   A interface do Apache Zeppelin é aberta em uma nova guia.
4. Na página Bem-vindo ao Zeppelin!, selecione Anotação do Zeppelin.
5. Na página Anotação do Zeppelin, insira a seguinte consulta em uma nova anotação:

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

Selecione o ícone de execução.

Depois de um curto período de tempo, a anotação exibe dados do Kinesis Data Streams.

Para abrir o painel do Apache Flink para que seu aplicativo visualize aspectos operacionais, selecione TRABALHO FLINK. Para obter mais informações sobre o painel do Flink, consulte o painel do Apache Flink no Guia do desenvolvedor do Managed Service for Apache Flink.

Para obter mais exemplos de consultas SQL do Flink Streaming, veja Consultas na documentação do Apache Flink.

Criar um bloco de anotações do Studio com o Amazon MSK

Este tutorial descreve como criar um bloco de anotações do Studio que usa um cluster do Amazon MSK como fonte.

Este tutorial contém as seguintes seções:

- Configuração
- Adicione um gateway NAT à sua VPC
- Criar uma conexão AWS Glue e tabela
- Crie um bloco de anotações do Studio com o Amazon MSK
- Envie dados para seu cluster Amazon MSK
- Teste seu bloco de anotações do Studio

Configuração

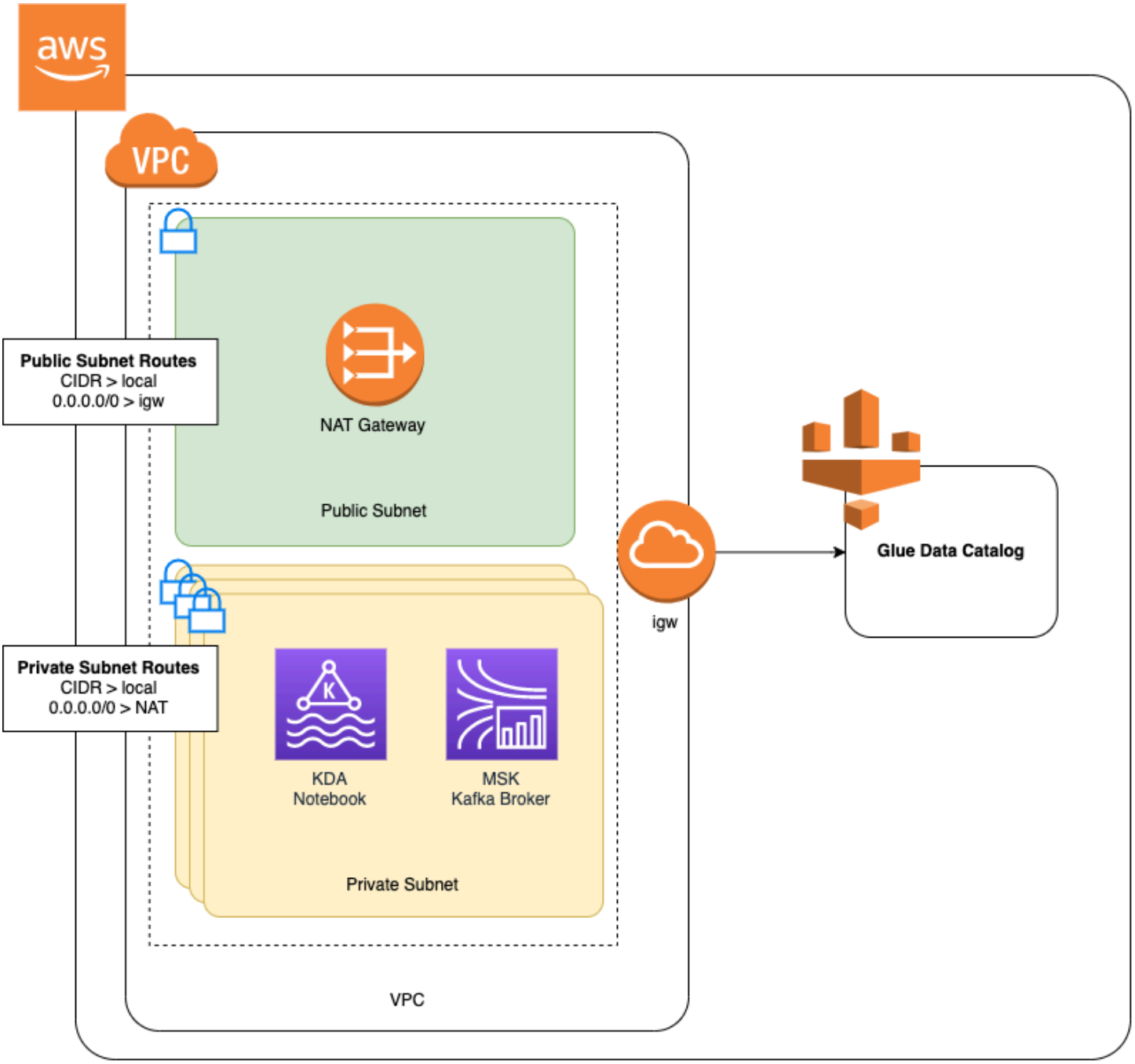
Para este tutorial, você precisa de um cluster do Amazon MSK que permita o acesso em texto sem formatação. Se você ainda não tiver um cluster Amazon MSK configurado, siga o tutorial Conceitos básicos do uso do Amazon MSK para criar uma Amazon VPC, um cluster Amazon MSK, um tópico e uma instância cliente do Amazon EC2.

Ao seguir o tutorial, faça o seguinte:

- Na Etapa 3: Criar um cluster Amazon MSK, na etapa 4, altere o valor `ClientBroker` de `TLS` para **`PLAINTEXT`**.

Adicione um gateway NAT à sua VPC

Se você criou um cluster Amazon MSK seguindo o tutorial Conceitos básicos do uso do Amazon MSK, ou se sua Amazon VPC existente ainda não tem um gateway NAT para suas sub-redes privadas, você deve adicionar um gateway NAT à sua Amazon VPC. O diagrama a seguir mostra a arquitetura.



Para criar um gateway NAT para sua Amazon VPC, faça o seguinte:

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Selecione Gateways NAT na barra de navegação à esquerda.
3. Na página Gateways NATs, escolha Criar gateway NAT.
4. Na página Criar gateway NAT, entre os valores a seguir:

Nome - opcional	**`ZeppelinGateway`**
Sub-rede	AWSKafkaTutorialSubnet1
ID de alocação do IP elástico	Choose an available Elastic IP. If there are no Elastic IPs available, choose Alocar IP elástico, and then choose the Elasic IP that the console creates.

Escolha Criar um gateway NAT.

5. Na barra de navegação à esquerda, escolha Tabelas de rotas.
6. Escolha Criar tabela de rotas.
7. Na página Criar tabela de rotas, forneça as seguintes informações:
   - Nome da tag: **`ZeppelinRouteTable`**
   - VPC: escolha sua VPC (por exemplo, AWSKafkaTutorialVPC).

   Escolha Criar.
8. Na lista de tabelas de rotas, selecione ZeppelinRouteTable. Escolha a guia Rotas e selecione Editar rotas.
9. Na página Editar rotas, selecione Adicionar rota.
10. Em Para Destino, insira **`0.0.0.0/0`**. Para Target, escolha gateway NAT, ZeppelinGateway. Selecione Salvar rotas. Escolha Fechar.
11. Na página Tabelas de rotas, com o ZeppelinRouteTable selecionado, escolha a guia Associações de sub-rede. Selecione Editar associações de sub-rede.
12. Na página Editar associações de sub-rede, selecione AWSKafkaTutorialSubnet2 e AWSKafkaTutorialSubnet3. Escolha Salvar.

Criar uma conexão AWS Glue e tabela

Seu bloco de anotações do Studio usa um banco de dados AWS Glue para metadados sobre sua fonte de dados Amazon MSK. Nesta seção, você cria uma conexão AWS Glue que descreve como

acessar seu cluster Amazon MSK e uma tabela AWS Glue que descreve como apresentar os dados em sua fonte de dados para clientes como seu bloco de anotações do Studio.

Criar uma conexão

1. Faça login no AWS Management Console e abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Se você ainda não tiver um banco de dados AWS Glue, selecione Bancos de dados na barra de navegação à esquerda. Selecione Adicionar banco de dados. Na janela Adicionar banco de dados, insira **`default`** para Nome do banco de dados. Escolha Criar.
3. Selecione Conexões na barra de navegação à esquerda. Selecione Adicionar conexão.
4. Na janela Adicionar conexão, forneça os seguintes valores:

   - Em Nome da conexão, insira **`ZeppelinConnection`**.
   - Em Tipo de conexão, escolha Kafka.
   - Para URLs do servidor bootstrap Kafka, forneça a sequência de caracteres do corretor bootstrap para seu cluster. Você pode obter os corretores de bootstrap no console do MSK ou digitando o seguinte comando da CLI:

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

   - Desmarque a caixa de seleção Exigir conexão SSL.

   Escolha Próximo.
5. Na página VPC, forneça os valores a seguir:

   - Para VPC, escolha o nome da sua VPC (por exemplo, AWSKafkaTutorialVPC.)
   - Em Sub-rede, escolha AWSKafkaTutorialSubnet2.
   - Em Grupos de segurança, escolha todos os grupos disponíveis.

   Escolha Próximo.
6. Na página Propriedades da conexão / Acesso à conexão, selecione Concluir.

Criar uma tabela

> **Note**
>
> Você pode criar manualmente a tabela conforme descrito nas etapas a seguir ou usar o código do conector de criação de tabela para o Managed Service for Apache Flink em seu bloco de anotações no Apache Zeppelin para criar sua tabela por meio de uma instrução DDL. Em seguida, você pode fazer o check-in no AWS Glue para garantir que a tabela foi criada corretamente.

1. Na barra de navegação à esquerda, selecione Tabelas. Na página Tabelas, selecione Adicionar tabelas, Adicionar tabela manualmente.
2. Na página Configurar propriedades da tabela, insira **`stock`** para o Nome da tabela. Certifique-se de selecionar o banco de dados que você criou anteriormente. Escolha Próximo.
3. Na página Adicionar um armazenamento de dados, selecione Kafka. Para o Nome do tópico, insira o nome do seu tópico (por exemplo, AWSKafkaTutorialTopic). Em Conexão, escolha ZeppelinConnection.
4. Na página Classificação, selecione JSON. Escolha Próximo.
5. Na página Definir um esquema, selecione Adicionar coluna para adicionar uma coluna. Adicione colunas com as seguintes propriedades:

Nome da coluna	Tipo de dados
`marcador de tempo`	**`string`**
`preço`	**`double`**

   Escolha Próximo.
6. Na próxima página, verifique suas configurações e selecione Concluir.
7. Selecione a tabela recém-criada na lista de tabelas.
8. Selecione Editar tabela e adicione uma propriedade com a chave `managed-flink.proctime` e o valor `proctime`.
9. Escolha Aplicar.

Crie um bloco de anotações do Studio com o Amazon MSK

Agora que você criou os recursos que o seu aplicativo usa, crie seu bloco de anotações do Studio.

Você pode criar seu aplicativo usando AWS Management Console ou AWS CLI.

- Crie um bloco de anotações do Studio usando AWS Management Console
- Crie um bloco de anotações do Studio usando AWS CLI

> **Note**
>
> Você também pode criar um bloco de anotações do Studio a partir do console Amazon MSK escolhendo um cluster existente e, em seguida, escolhendo Processar dados em tempo real.

Crie um bloco de anotações do Studio usando AWS Management Console

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Na página Aplicativos do Managed Service for Apache Flink, selecione a guia Studio. Selecione Criar bloco de anotações do Studio.

   > **Note**
   >
   > Para criar um bloco de anotações do Studio a partir dos consoles Amazon MSK ou Kinesis Data Streams, selecione seu cluster Amazon MSK de entrada ou fluxo de dados do Kinesis e escolha Processar dados em tempo real.

3. Na página Criar instância de bloco de anotações, forneça as seguintes informações:

   - Insira **`MyNotebook`** para Nome do bloco de anotações do Studio.
   - Selecione o padrão para o banco de dados AWS Glue.

   Selecione Criar bloco de anotações do Studio.
4. Na página MyNotebook, selecione a guia Configuração. Na seção Redes, selecione Editar.
5. Na página Editar rede para MyNotebook, selecione Configuração de VPC com base no cluster Amazon MSK. Selecione seu cluster Amazon MSK para Cluster Amazon MSK. Escolha Salvar alterações.

6. Na página MyNotebook, selecione Executar. Aguarde até que o Status mostre Em execução.

Crie um bloco de anotações do Studio usando AWS CLI

Para criar seu bloco de anotações do Studio usando AWS CLI, faça o seguinte:

1. Verifique se você tem as informações a seguir. Você precisa desses valores para criar seu aplicativo.
   - O ID da sua conta.
   - Os IDs de sub-rede e o ID do grupo de segurança da Amazon VPC que contém o cluster Amazon MSK.
2. Crie um arquivo chamado `create.json` com o conteúdo a seguir. Substitua os valores de espaço reservado por suas próprias informações.

```
{
    "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 criar o seu aplicativo, execute o comando a seguir:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
```

4. Quando o comando for concluído, você deverá ver um resultado semelhante ao apresentado a seguir, mostrando os detalhes do seu novo bloco de anotações do Studio:

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

5. Execute o comando a seguir para iniciar o aplicativo. Substitua o valor de amostra pelo ID de sua conta.

```
aws kinesisanalyticsv2 start-application --application-arn
 arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

Envie dados para seu cluster Amazon MSK

Nesta seção, você executa um script Python em seu cliente Amazon EC2 para enviar dados para sua fonte de dados Amazon MSK.

1. Conecte-se ao seu cliente do Amazon EC2.
2. Execute os comandos a seguir para instalar o Python versão 3, o Pip e o pacote Kafka para Python e confirme as ações:

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

```
pip install kafka-python
```

3. Configure AWS CLI na máquina do cliente inserindo o comando a seguir:

```
aws configure
```

   Forneça as credenciais da sua conta e **us-east-1** para `region`.

4. Crie um arquivo chamado `stock.py` com o conteúdo a seguir. Substitua o valor de amostra pela sequência de caracteres do corretor bootstrap do seu cluster Amazon MSK e atualize o nome do tópico se o tópico não for 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. Execute o script com o comando a seguir:

```
$ python3 stock.py
```

6. Deixe o script em execução enquanto você conclui a seção a seguir.

Teste seu bloco de anotações do Studio

Nesta seção, você usa seu bloco de anotações do Studio para consultar dados do seu cluster Amazon MSK.

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Na página Aplicativos do Managed Service for Apache Flink, escolha a guia Bloco de anotações do Studio. Selecione MyNotebook.
3. Na página MyNotebook, selecione Abrir no Apache Zeppelin.

   A interface do Apache Zeppelin é aberta em uma nova guia.
4. Na página Bem-vindo ao Zeppelin!, selecione Nova anotação do Zeppelin.
5. Na página Anotação do Zeppelin, insira a seguinte consulta em uma nova anotação:

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

   Selecione o ícone de execução.

   O aplicativo exibe dados do cluster Amazon MSK.

Para abrir o painel do Apache Flink para que seu aplicativo visualize aspectos operacionais, selecione TRABALHO FLINK. Para obter mais informações sobre o painel do Flink, consulte o Painel do Apache Flink no Guia do desenvolvedor do Managed Service for Apache Flink.

Para obter mais exemplos de consultas SQL do Flink Streaming, veja Consultas na Documentação do Apache Flink.

Limpar seu aplicativo e os recursos dependentes

Exclua seu bloco de anotações do Studio

1. Abra o console do Managed Service for Apache Flink.
2. Selecione MyNotebook.
3. Selecione Ações e Excluir.

Exclua seu banco de dados AWS Glue e conexão

1. Abra o console do AWS Glue em https://console.aws.amazon.com/glue/.
2. Selecione Bancos de dados na barra de navegação à esquerda. Marque a caixa de seleção ao lado de Padrão para selecioná-la. Selecione Ação, Excluir banco de dados. Confirme a seleção.
3. Selecione Conexões na barra de navegação à esquerda. Marque a caixa de seleção ao lado de ZeppelInConnection para selecioná-la. Selecione Ação, Excluir conexão. Confirme a seleção.

Exclua a função e a política do perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No menu de navegação à esquerda, selecione Funções.
3. Use a barra de pesquisa para procurar a função ZeppelinRole.
4. Selecione a função ZeppelinRole. Selecione Excluir função. Confirme a exclusão.

Excluir o grupo de logs do CloudWatch

O console cria um grupo de logs e um fluxo de logs do CloudWatch para você quando você cria seu aplicativo usando o console. Você não tem um grupo de logs e um fluxo de logs se tiver criado seu aplicativo usando o AWS CLI.

1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. Selecione Grupos de log na barra de navegação à esquerda.
3. Selecione o grupo de logs /aws/kinesis-analytics/MyApplication.
4. Selecione Actions (Ações), Delete log group(s) (Excluir grupo(s) de log). Confirme a exclusão.

Limpar os recursos do Kinesis Data Streams

Para excluir o seu fluxo do Kinesis, abra o console do Kinesis Data Streams, selecione seu fluxo do Kinesis e selecione Ações, Excluir.

Limpar os recursos do MSK

Siga as etapas nesta seção se você criou um cluster Amazon MSK para este tutorial. Esta seção tem instruções para limpar sua instância cliente Amazon EC2, Amazon VPC e cluster Amazon MSK.

Excluir seu cluster do Amazon MSK

Siga estas etapas se você criou um cluster Amazon MSK para este tutorial.

1. Abra o console do Amazon MSK em https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. Selecione AWSKafka TutorialCluster. Selecione Delete (Excluir). Insira **`delete`** na janela que aparece e confirme sua seleção.

Encerrar sua instância cliente

Siga estas etapas se você criou uma instância cliente do Amazon EC2 para este tutorial.

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Selecione Instâncias na barra de navegação à esquerda.
3. Marque a caixa de seleção ao lado do ZeppelinClient para selecioná-la.
4. Selecione Estado da instância e Encerrar instância.

Excluir sua Amazon VPC

Siga estas etapas se você criou uma Amazon VPC para este tutorial.

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Selecione Interfaces de rede na barra de navegação à esquerda.
3. Insira a sua ID de VPC na barra de pesquisa e pressione Enter para pesquisar.
4. Marque a caixa de seleção no cabeçalho da tabela para selecionar todas as interfaces de rede exibidas.

5. Clique em Actions (Ações) e em Detach (Desanexar). Na janela exibida, selecione Ativar em Forçar desanexação. Selecione Desanexar e aguarde até que todas as interfaces de rede atinjam o status Disponível.
6. Marque a caixa de seleção no cabeçalho da tabela para selecionar novamente todas as interfaces de rede exibidas.
7. Selecione Actions, Delete. Confirme a ação.
8. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
9. Selecione AWSKafkaTutorialVPC. Selecione Actions, Excluir VPC. Entre **`delete`** e confirme a exclusão.

Tutorial: implantação como um aplicativo com estado durável

O tutorial a seguir demonstra como implantar um Studio notebook como um aplicativo Managed Service for Apache Flink com estado durável.

Este tutorial contém as seguintes seções:

- Configuração
- Implemente um aplicativo com estado durável usando o AWS Management Console
- Implemente um aplicativo com estado durável usando o AWS CLI

Configuração

Crie um novo Studio notebook seguindo o Tutorial de criação de um bloco de anotações Studio, usando o fluxo de dados Kinesis ou o Amazon MSK. Dê um nome ao Studio notebook `ExampleTestDeploy`.

Implemente um aplicativo com estado durável usando o AWS Management Console

1. Adicione um local de bucket do S3 onde deseja que o código empacotado seja armazenado em Localização do código do aplicativo - opcional no console. Isso habilita as etapas para implantar e executar seu aplicativo diretamente do notebook.
2. Adicione as permissões necessárias à função do aplicativo para habilitar a função que você está usando para ler e gravar em um bucket do Amazon S3 e para iniciar um aplicativo Managed Service for Apache Flink:

    - AmazonS3FullAccess

- Amazonmanaged-flinkFullAccess
- Acesso às suas fontes, destinos e VPCs, conforme aplicável. Para obter mais informações, consulte Permissões do IAM para notebooks Studio.

3. Use o seguinte código de exemplo:

```
%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. Com o lançamento desse atributo, você verá uma nova lista suspensa no canto superior direito de cada nota em seu notebook com o nome do notebook. Você pode fazer o seguinte:

- Veja as configurações do Studio notebook no AWS Management Console.
- Crie seu Zeppelin Note e exporte-o para o Amazon S3. Nesse ponto, forneça um nome para seu aplicativo e selecione Criar e exportar. Você receberá uma notificação quando a exportação for concluída.
- Se precisar, você pode visualizar e executar quaisquer testes adicionais no executável no Amazon S3.
- Quando a compilação estiver concluída, você poderá implantar seu código como um aplicativo de transmissão do Kinesis com estado durável e escalabilidade automática.
- Use o menu suspenso e selecione Implantar o Zeppelin Note como aplicativo de transmissão do Kinesis. Revise o nome do aplicativo e selecione Implantar via AWS console.
- Isso levará você à página AWS Management Console para a criação de um aplicativo Managed Service for Apache Flink. Observe que o nome do aplicativo, o paralelismo, a localização do código, as funções padrão do Glue DB, VPC (se aplicável) e IAM foram pré-preenchidas. Valide se as funções do IAM têm as permissões necessárias para suas fontes e

destinos. Os snapshots são habilitados por padrão para um gerenciamento durável do estado do aplicativo.

- Selecione Criar aplicativo.
- Você pode selecionar Configurar e modificar qualquer configuração e selecionar Executar para iniciar seu aplicativo de transmissão.

Implemente um aplicativo com estado durável usando o AWS CLI

Para implantar um aplicativo usando o AWS CLI, você deve atualizar seu AWS CLI para usar o modelo de serviço fornecido com as informações do Beta 2. Para obter informações sobre como usar o modelo de serviço atualizado, consulte Configuração.

O código de exemplo a seguir cria um novo Studio notebook:

```
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
```

O código de exemplo a seguir inicia um Studio notebook:

```
aws kinesisanalyticsv2 start-application \
    --application-name <app-name> \
    --region us-east-1 \
    --no-verify-ssl
```

O código a seguir retorna a URL da página do Apache Zeppelin notebook de um aplicativo:

```
aws kinesisanalyticsv2 create-application-presigned-url \
    --application-name <app-name> \
    --url-type ZEPPELIN_UI_URL \

    --region us-east-1 \
    --no-verify-ssl
```

Exemplos

Os exemplos de consultas a seguir demonstram como analisar dados usando consultas de janela em um notebook Studio.

- Criação de tabelas com o Amazon MSK/Apache Kafka
- Criar tabelas com o Kinesis
- Janela em cascata
- Janela deslizante
- SQL interativo

- BlackHole Conector SQL
- Gerador de dados
- Scala interativa
- Python interativo
- Interativos Python, SQL e Scala
- Fluxo de dados do Kinesis entre contas

Para obter informações sobre as configurações de consulta SQL do Apache Flink, consulte Flink on Zeppelin Notebooks for Interactive Data Analysis.

Para visualizar seu aplicativo no painel do Apache Flink, selecione FLINK JOB na página Zeppelin Note do seu aplicativo.

Para obter mais informações sobre consultas de janela, consulte Windows na documentação do Apache Flink.

Para obter mais exemplos de consultas SQL do Apache Flink Streaming, consulte Consultas na documentação do Apache Flink.

Criação de tabelas com o Amazon MSK/Apache Kafka

Você pode usar o conector Amazon MSK Flink com o Managed Service for Apache Flink Studio para autenticar sua conexão com autenticação de texto simples, SSL ou IAM. Crie suas tabelas usando as propriedades específicas de acordo com seus 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'
);
```

Você pode combiná-las com outras propriedades no Apache Kafka SQL Connector.

Criar tabelas com o Kinesis

No exemplo a seguir, você cria uma tabela usando o 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 obter mais informações sobre outras propriedades que você pode usar, consulte Amazon Kinesis Data Streams SQL Connector.

Janela em cascata

A consulta SQL do Flink Streaming a seguir seleciona o preço mais alto em cada janela em cascata de cinco segundos da tabela `ZeppelinTopic`:

```
%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)
```

Janela deslizante

A consulta SQL do Flink Streaming a seguir seleciona o preço mais alto em cada janela deslizante de cinco segundos da tabela `ZeppelinTopic`:

```
%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 interativo

Este exemplo imprime o tempo máximo do evento e o tempo de processamento e a soma dos valores da tabela de valores-chave. Certifique-se de ter o exemplo de script de geração de dados da execução the section called “Gerador de dados”. Para experimentar outras consultas SQL, como filtragem e junções em seu notebook Studio, consulte a documentação do Apache Flink: Consultas na documentação do 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

O conector BlackHole SQL não exige que você crie um stream de dados do Kinesis ou um cluster Amazon MSK para testar suas consultas. Para obter informações sobre o conector BlackHole SQL,

consulte Conector BlackHole SQL na documentação do Apache Flink. Neste exemplo, o catálogo padrão é um catálogo na memória.

```
%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
```

Gerador de dados

Este exemplo usa o Scala para gerar dados de amostra. Você pode usar esses dados de exemplo para testar várias consultas. Use a instrução criar tabela para criar a tabela de valores-chave.

```
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 interativa

Esta é a tradução em Scala do the section called “SQL interativo”. Para ver mais exemplos de Scala, consulte API de tabela na documentação do 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 interativo

Esta é a tradução em Python do the section called “SQL interativo”. Para ver mais exemplos de Python, consulte API de tabela na documentação do 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`
```

Interativos Python, SQL e Scala

Você pode usar qualquer combinação de SQL, Python e Scala em seu notebook para análise interativa. Em um notebook do Studio que você planeja implantar como um aplicativo com estado durável, você pode usar uma combinação de SQL e Scala. Este exemplo mostra as seções que são ignoradas e aquelas que são implantadas no aplicativo com estado durável.

```
%flink.ssql
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.ssql
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!")
```

Fluxo de dados do Kinesis entre contas

Para usar um fluxo de dados do Kinesis que está em uma conta diferente da conta que tem seu notebook Studio, crie uma função de execução de serviço na conta em que seu notebook Studio está sendo executado e uma política de confiança de função na conta que tem o fluxo de dados. Use `aws.credentials.provider`, `aws.credentials.role.arn` e `aws.credentials.role.sessionName` no conector Kinesis em sua instrução criar tabela DDL para criar uma tabela em relação ao fluxo de dados.

Use a seguinte função de execução de serviço para a conta do notebook Studio.

```
{
 "Sid": "AllowNotebookToAssumeRole",
 "Effect": "Allow",
 "Action": "sts:AssumeRole"
 "Resource": "*"
```

```
}
```

Use a política `AmazonKinesisFullAccess` e a seguinte política de confiança de função para a conta de fluxo de dados.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<accountID>:root"
 },
 "Action": "sts:AssumeRole",
 "Condition": {}
 }
 ]
}
```

Use o parágrafo a seguir para a instrução de criação de tabela.

```
%flink.ssql
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'
)
```

Solução de problemas

Esta seção contém informações sobre solução de problemas para notebooks Studio.

Interromper um aplicativo bloqueado

Para interromper um aplicativo que está preso em um estado transitório, chame a StopApplicationação com o `Force` parâmetro definido como. `true` Para obter mais informações, consulte Running Applications, no Guia do desenvolvedor do Managed Service for Apache Flink.

Implantar como um aplicativo com estado durável em uma VPC sem acesso à Internet

A deploy-as-application função Managed Service for Apache Flink Studio não oferece suporte a aplicativos VPC sem acesso à Internet. Recomendamos que você crie seu aplicativo no Studio e, em seguida, use o Managed Service for Apache Flink para criar manualmente um aplicativo Flink e selecionar o arquivo zip que você criou em seu Notebook.

As etapas a seguir descrevem essa abordagem:

1. Crie e exporte seu aplicativo Studio para o Amazon S3. Isso deve ser um arquivo zip.
2. Crie um aplicativo Managed Service for Apache Flink manualmente com um caminho de código referenciando a localização do arquivo zip no Amazon S3. Além disso, você precisará configurar o aplicativo com as seguintes variáveis `env` (2 `groupID`, 3 `var` no total):
3. kinesis.analytics.flink.run.options
    a. python: source/note.py
    b. arquivo jar: PythonApplicationDependencies lib/.jar
4. managed.deploy_as_app.options
    - DatabaseARN: *`<glue database ARN (Amazon Resource Name)>`*
5. Talvez seja necessário conceder permissões aos perfis do IAM do Managed Service for Apache Flink Studio e Managed Service for Apache Flink para os serviços que seu aplicativo usa. Você pode usar o mesmo perfil do IAM para os dois aplicativos.

Redução eploy-as-app do tamanho D e do tempo de construção

Os aplicativos Studio deploy-as-app for Python empacotam tudo o que está disponível no ambiente Python porque não podemos determinar quais bibliotecas você precisa. Isso pode resultar em um tamanho maior do que o necessário. deploy-as-app O procedimento a seguir demonstra como reduzir o tamanho do aplicativo deploy-as-app Python desinstalando dependências.

Se você estiver criando um aplicativo Python com deploy-as-app recursos do Studio, considere remover pacotes Python pré-instalados do sistema, caso seus aplicativos não dependam deles. Isso não só ajudará a reduzir o tamanho final do artefato para evitar a violação do limite de serviço para o tamanho do aplicativo, mas também melhorará o tempo de criação dos aplicativos com o deploy-as-app recurso.

Execute o comando a seguir para listar todos os pacotes Python instalados com seus respectivos tamanhos instalados e remover seletivamente os pacotes com tamanho 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**
>
> Para funcionar, o Flink Python exige o `apache-beam`. Você nunca deve remover esse pacote e suas dependências.

A seguir está a lista de pacotes Python pré-instalados no Studio V2 que podem ser considerados para remoção:

```
scipy
statsmodels
plotnine
seaborn
llvmlite
bokeh
pandas
matplotlib
botocore
boto3
numba
```

Para remover um pacote Python do caderno do Zeppelin:

1. Verifique se o aplicativo depende do pacote, ou de qualquer um dos pacotes que o consome, antes de removê-lo. Identifique os dependentes de um pacote usando pipdeptree.

2. Executar o seguinte comando para remover um pacote:

```
%flink.pyflink
!pip uninstall -y <package-to-remove>
```

3. Se você precisar recuperar um pacote removido por engano, execute o seguinte comando:

```
%flink.pyflink
!pip install <package-to-install>
```

Example Exemplo: remova o **`scipy`** pacote antes de implantar seu aplicativo deploy-as-app Python com o recurso.

1. Use `pipdeptree` para descobrir todos os consumidores do `scipy` e verificar se você pode remover `scipy` com segurança.

   - Instale a ferramenta por meio do caderno:

   ```
   %flink.pyflink
   !pip install pipdeptree
   ```

   - Obtenha a árvore de dependências revertida do `scipy` executando:

   ```
   %flink.pyflink
   !pip -r -p scipy
   ```

   Você verá um resultado semelhante ao seguinte (reduzido para concisão):

   ```
   ...
   ------------------------------------------------------------------------
   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. Inspecione cuidadosamente o uso de `seaborn`, `statsmodels` e `plotnine` em seus aplicativos. Se os aplicativos não dependerem do `scipy`, `seaborn`, `statemodels` ou `plotnine`, você poderá remover todos esses pacotes ou somente aqueles que os aplicativos não precisarem.

3. Remova o pacote executando:

```
!pip uninstall -y scipy plotnine seaborn statemodels
```

Cancelar trabalhos

Esta seção mostra como cancelar trabalhos do Apache Flink que você não pode acessar no Apache Zeppelin. Se você quiser cancelar esse trabalho, acesse o painel do Apache Flink, copie o ID do trabalho e use-o em um dos exemplos a seguir.

Para cancelar um único trabalho:

```
%flink.pyflink
import requests

requests.patch("https://zeppelin-flink:8082/jobs/[job_id]", verify=False)
```

Para cancelar todos os trabalhos em execução:

```
%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 os trabalhos:

```
%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 o interpretador Apache Flink

Para reiniciar o interpretador Apache Flink em seu notebook Studio

1. Selecione Configuração no canto superior direito da tela.
2. Selecione Intérprete.
3. Selecione reiniciar e, em seguida, OK.

Apêndice: Criar políticas personalizadas do IAM

Normalmente, você usa políticas gerenciadas do IAM para permitir que seu aplicativo acesse recursos dependentes. Se precisar de um controle mais preciso sobre as permissões do seu aplicativo, você pode usar uma política personalizada do IAM. Esta seção contém exemplos de políticas personalizadas do IAM.

> **Note**
>
> Nos exemplos de políticas a seguir, substitua o texto do espaço reservado pelos valores do seu aplicativo.

Este tópico contém as seguintes seções:

- AWS Glue
- CloudWatch Registros
- Streams do Kinesis
- Clusters do Amazon MSK

AWS Glue

O exemplo da seguir e política concede permissões de acesso a base de dados 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

A política a seguir concede permissões para acessar CloudWatch os 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**
>
> Se você criar seu aplicativo usando o console, o console adicionará as políticas necessárias para acessar CloudWatch Logs à sua função de aplicativo.

Streams do Kinesis

Seu aplicativo pode usar um Kinesis Stream como origem ou destino. Seu aplicativo precisa de permissões de leitura para ler de um stream de origem e permissões de gravação para gravar em um stream de destino.

A política a seguir concede permissões para leitura de um Kinesis Stream usado como fonte:

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

A política a seguir concede permissões para gravar em um Kinesis Stream usado como destino:

```
{
    "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>"
        }
```

```
    ]
}
```

Se seu aplicativo acessar um stream criptografado do Kinesis, você deverá conceder permissões adicionais para acessar o stream e a chave de criptografia do stream.

A política a seguir concede permissões para acessar um stream de origem criptografado e a chave de criptografia do stream:

```
{
      "Sid": "ReadEncryptedKinesisStreamSource",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "<inputStreamKeyArn>"
      ]
    }
    ,
```

A política a seguir concede permissões para acessar um stream de destino criptografado e a chave de criptografia do stream:

```
{
      "Sid": "WriteEncryptedKinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "<outputStreamKeyArn>"
      ]
    }
```

Clusters do Amazon MSK

Para conceder acesso a um cluster Amazon MSK, você concede acesso à VPC do cluster. Para exemplos de políticas para acessar uma Amazon VPC, consulte Permissões do aplicativo VPC.

Introdução ao Amazon Managed Service para Apache Flink (DataStream API)

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink e da API. DataStream Ela descreve as opções disponíveis para criar e testar seus aplicativos. Ela também fornece instruções para instalar as ferramentas necessárias para concluir os tutoriais deste guia e criar seu primeiro aplicativo.

Tópicos

Componentes do aplicativo do Managed Service for Apache Flink

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java/ Apache Maven ou Scala que processa a entrada e produz a saída usando o runtime do Apache Flink.

O Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.
- Fonte: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um bucket do Amazon S3 etc. Para ter mais informações, consulte Origens.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Operadores da API do DataStream.

- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis Data Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Coletores.

Depois de criar, compilar e empacotar o código da aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon Simple Storage Service (Amazon S3). Em seguida, crie um aplicativo do Managed Service for Apache Flink. Você passa na localização do pacote de código, um fluxo de dados do Kinesis como fonte de dados de fluxo e, normalmente, um local de fluxo ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos para concluir os exercícios

Para concluir as etapas neste guia, você deve ter o seguinte:

- Java Development Kit (JDK) versão 11. Defina a variável do ambiente`JAVA_HOME` para apontar para o local de instalação do JDK.
- Recomendamos que você use um ambiente de desenvolvimento (como Eclipse Java Neon ou IntelliJ Idea) para desenvolver e compilar seu aplicativo.
- Cliente do Git. Instale o cliente do Git se você ainda não tiver feito isso.
- Apache Maven Compiler Plugin. Maven deve estar em seu caminho de trabalho. Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

Para começar a usar, vá até Etapa 1: Configurar uma conta da AWS e criar um usuário administrador.

Etapa 1: Configurar uma conta da AWS e criar um usuário administrador

Antes de usar o Managed Service for Apache Flink pela primeira vez, execute as seguintes tarefas:

Cadastrar-se em uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas a seguir para criar uma.

Para se cadastrar em uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

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

   Quando você se cadastra em uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua acesso administrativo a um usuário administrativo e use somente o usuário raiz para realizar as tarefas que exigem acesso do usuário raiz.

A AWS envia um e-mail de confirmação depois que o processo de cadastramento é concluído. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário administrativo

Depois de se inscrever em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o AWS IAM Identity Center e crie um usuário administrativo para não usar o usuário raiz em tarefas cotidianas.

Proteger seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Console como o proprietário da conta ao escolher a opção Usuário raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, digite sua senha.

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

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

Criar um usuário administrativo

1. Habilitar o IAM Identity Center.

Para obter instruções, consulte Enabling AWS IAM Identity Center no Manual do Usuário do AWS IAM Identity Center.

2. No Centro de Identidade do IAM, conceda acesso administrativo a um usuário administrativo.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configure user access with the default Diretório do Centro de Identidade do IAM no Manual do Usuário do AWS IAM Identity Center.

Login como usuário administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda com o login utilizando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Conceder acesso programático

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

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

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Configuração da AWS CLI para usar o AWS IAM Identity Center no AWS Command Line Interface Guia do usuário da .

Qual usuário precisa de acesso programático?	Para	Por
		• Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções em Como usar credenciais temporárias com recursos da AWS no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Próxima etapa

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura a AWS CLI para uso com o Managed Service for Apache Flink.

> **Note**
>
> Os exercícios de conceitos básicos neste guia pressupõem que você esteja usando credenciais de administrador (`adminuser`) em sua conta para executar as operações.

> **Note**
>
> Se você já tiver a AWS CLI instalada, pode ser necessário atualizá-la para obter as funcionalidades mais recentes. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface. Para verificar a versão da AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios neste tutorial requerem a seguinte versão da AWS CLI ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar a AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Guia do usuário do AWS Command Line Interface:
   - Instalar a AWS Command Line Interface
   - Configurar a AWS CLI
2. Adicione um perfil nomeado para o usuário administrador no arquivo AWS CLI `config`. Você pode usar esse perfil ao executar os comandos da AWS CLI. Para obter mais informações sobre perfis nomeados, consulte Perfis nomeados no Guia do usuário da 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 obter uma lista das regiões da AWS disponíveis, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

> **Note**
>
> O código e os comandos de exemplo neste tutorial usam a região Oeste dos EUA (Oregon). Para usar uma região diferente, altere a região no código e nos comandos deste tutorial para a região que você deseja usar.

3. Verifique a configuração digitando o seguinte comando no prompt de comando:

```
aws help
```

Depois de configurar uma AWS conta e aAWS CLI, você pode tentar o próximo exercício, no qual você configura um aplicativo de amostra e testa a end-to-end configuração.

Próxima etapa

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Neste exercício, você cria um aplicativo Managed Service for Apache Flink com fluxos de dados como origem e coletor.

Esta seção contém as seguintes etapas:

- Criar dois fluxos de dados do Amazon Kinesis Data Streams
- Gravação de registros de amostra no fluxo de entrada
- Baixar e examinar o código Java Apache Flink Streaming
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Próxima etapa

Criar dois fluxos de dados do Amazon Kinesis Data Streams

Antes de criar um aplicativo do Managed Service for Apache Flink para este exercício, crie dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`). O aplicativo usa esses fluxos para os fluxos de origem e de destino do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou o comando da AWS CLI a seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro fluxo (`ExampleInputStream`), use o comando `create-stream` AWS CLI do Amazon Kinesis a seguir.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Para criar o segundo fluxo que o aplicativo usa para gravar a saída, execute o mesmo comando, alterando o nome da transmissão para `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Mais adiante neste tutorial, você executa o script `stock.py` para enviar dados para o aplicativo.

```
$ python stock.py
```

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo Project Object Model (pom.xml) contém informações sobre a configuração e as dependências da aplicação, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.java` contém o método `main` que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Seu aplicativo cria conectores de origem e de destino para acessar recursos externos usando um objeto `StreamExecutionEnvironment`.
- O aplicativo cria conectores de origem e de destino usando propriedades estáticas. Para usar as propriedades do aplicativo dinâmico, use os métodos `createSinkFromApplicationProperties` e `createSourceFromApplicationProperties` para criar os conectores. Esses métodos leem as propriedades do aplicativo para configurar os conectores.

  Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compilar o código do aplicativo

Nesta seção, você usa o compilador do Apache Maven para criar o código Java para o aplicativo. Para obter informações sobre como instalar o Apache Maven e o Java Development Kit (JDK), consulte Pré-requisitos para concluir os exercícios.

Para compilar o código do aplicativo

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Há duas formas de compilar e empacotar o código:

- Use a ferramenta de linha de comando do Maven. Crie seu arquivo JAR executando o seguinte comando no diretório que contém o arquivo `pom.xml`:

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

- Use o ambiente de desenvolvimento. Consulte a documentação de seu ambiente de desenvolvimento para obter mais detalhes.

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

Você pode carregar o pacote como um arquivo JAR, ou pode compactar o pacote e carregá-lo como um arquivo ZIP. Se você criar seu aplicativo usando a AWS CLI, especifique o tipo de conteúdo de código (JAR ou ZIP).

2. Se houver erros durante a compilação, verifique se sua variável de ambiente `JAVA_HOME` está definida corretamente.

Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

faça o upload do código Java Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon Simple Storage Service (Amazon S3) e faz upload do código do aplicativo.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira **`ka-app-code-<username>`** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).

5. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
6. Selecione Criar bucket.
7. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior. Selecione Next (Próximo).
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI.

> **Note**
>
> Quando você cria o aplicativo usando o console, seus recursos AWS Identity and Access Management (IAM) e do Amazon CloudWatch Logs são criados para você. Quando cria o aplicativo usando a AWS CLI, você cria esses recursos separadamente.

Tópicos

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.

3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*`012345678901`*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, escolha Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Properties (Propriedades), Group ID (ID do grupo), insira **ProducerConfigProperties**.
5. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**

ID do grupo	Chave	Valor
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
7. Para CloudWatch registrar, marque a caixa de seleção Ativar.
8. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualize o aplicativo

Usando o console, você pode atualizar configurações do aplicativo, como as propriedades do aplicativo, as configurações de monitoramento e a localização ou o nome do arquivo JAR do aplicativo. Você também pode recarregar o JAR do aplicativo do bucket do Amazon S3 se precisar atualizar o código do aplicativo.

Na MyApplicationpágina, escolha Configurar. Atualize as configurações do aplicativo e selecione Update (Atualizar).

Criar e executar o aplicativo (AWS CLI)

Nesta seção, você usa a AWS CLI para criar e executar o aplicativo Flink do Managed Service for Apache Flink. O Managed Service for Apache Flink usa o comando `kinesisanalyticsv2` AWS CLI para criar e interagir com aplicativos Managed Service for Apache Flink.

Criar uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação `read` no fluxo de origem, e outra que concede permissões para ações `write` no fluxo de destino. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua *`username`* pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos Nomes de recurso da Amazon (ARNs) (*`012345678901`*) pelo ID da conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

> **Note**
>
> Para acessar outros serviços da Amazon, você pode usar o AWS SDK for Java. O Managed Service for Apache Flink define automaticamente as credenciais exigidas pelo SDK como as credenciais do perfil do IAM associado a seu aplicativo. Não é necessária nenhuma etapa adicional.

Criar uma perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles (Funções) e Create Role (Criar função).
3. Em Selecionar tipo de identidade de confiança, selecione Serviço da AWS. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis. Em Select your use case (Selecionar seu caso de uso), selecione Kinesis Analytics.

   Selecione Next: Permissions (Próximo: permissões).
4. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
5. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
6. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, the section called “Criar uma política de permissões”.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).

   b. Selecione Attach Policies.

   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).

   d. Escolha a ReadSourceStreamWriteSinkStream política AK e escolha Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo do Managed Service for Apache Flink

1. Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (*username*) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (*012345678901*) na função de execução do serviço pelo ID da conta.

```
{
    "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. Execute a ação `CreateApplication` com a solicitação anterior para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação `StartApplication` para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação `StopApplication` para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Execute a ação `StopApplication` com a seguinte solicitação para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consultethe section called “Configurando o log”.

Atualizar propriedades do ambiente

Nesta seção, você usa a ação `UpdateApplication` para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação [UpdateApplication](#) com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação [UpdateApplication](#)AWS CLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como [ativar ou desativar](#) o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (*<username>*) com o sufixo que

você selecionou na seção the section called “Criar dois fluxos de dados do 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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Próxima etapa

Etapa 4: limpar os recursos AWS

Etapa 4: limpar os recursos AWS

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial Introdução.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Excluir seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos
- Próxima etapa

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.

2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seus fluxos de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.

3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Próxima etapa

Etapa 5: próximas etapas

Etapa 5: próximas etapas

Agora que você criou e executou um aplicativo básico do Managed Service for Apache Flink, consulte os seguintes recursos para obter soluções mais avançadas de Managed Service for Apache Flink.

- A solução de dados de transmissão para o Amazon Kinesis da AWS: A solução de dados de transmissão para o Amazon Kinesis da AWS configura automaticamente os serviços da AWS necessários para capturar, armazenar, processar e entregar dados de transmissão com facilidade. A solução oferece várias opções para resolver casos de uso de dados de transmissão. A opção Managed Service for Apache Flink fornece um exemplo de ETL de end-to-end streaming que demonstra um aplicativo do mundo real que executa operações analíticas em dados simulados de táxi de Nova York. A solução configura todos os AWS recursos necessários, como funções e políticas do IAM, um CloudWatch painel e CloudWatch alarmes.
- Solução de transmissão de dados para o Amazon MSK da AWS: A solução de transmissão de dados para o Amazon MSK da AWS fornece modelos de AWS CloudFormation onde os dados fluem por produtores, armazenamento de transmissão, consumidores e destinos.
- Clickstream Lab com Apache Flink e Apache Kafka: um laboratório completo para casos de uso de clickstream usando Amazon Managed Streaming for Apache Kafka para armazenamento de transmissão e aplicativos Managed Service for Apache Flink for Apache Flink para processamento de fluxos.
- Workshop do Amazon Managed Service para Apache Flink: Neste workshop, você cria uma arquitetura de end-to-end streaming para ingerir, analisar e visualizar dados de streaming quase em tempo real. Você decidiu melhorar as operações de uma empresa de táxi na cidade de Nova York. Você analisa os dados de telemetria de uma frota de táxis na cidade de Nova York quase em tempo real para otimizar as operações da frota.
- Managed Service for Apache Flink: exemplos: Esta seção deste Guia do desenvolvedor fornece exemplos de como criar e trabalhar com aplicativos no Managed Service for Apache Flink. Eles

incluem exemplos de código e step-by-step instruções para ajudá-lo a criar serviços gerenciados para aplicativos Apache Flink e testar seus resultados.

- Conheça o Flink: treinamento prático: Treinamento introdutório oficial do Apache Flink que ajuda você a começar a escrever ETL de transmissão escalável, análises e aplicativos orientados a eventos.

> **Note**
>
> Esteja ciente de que o Managed Service for Apache Flink não é compatível com a versão Apache Flink (1.12) usada neste treinamento. Você pode usar o Flink 1.15.2 no Flink Managed Service para Apache Flink.

Conceitos básicos do Amazon Managed Service for Apache Flink (API de tabela)

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink e da API de tabela. Ela descreve as opções disponíveis para criar e testar seus aplicativos. Ela também fornece instruções para instalar as ferramentas necessárias para concluir os tutoriais deste guia e criar seu primeiro aplicativo.

Tópicos

Componentes do aplicativo do Managed Service for Apache Flink

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java/Apache Maven ou Scala que processa a entrada e produz a saída usando o runtime do Apache Flink.

O aplicativo do Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.
- Fonte da tabela: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um tópico do Amazon MSK ou similar. Para ter mais informações, consulte Fontes da API de tabela.
- Funções: o aplicativo processa dados usando uma ou mais funções. Uma função pode transformar, enriquecer ou agregar dados.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis, um fluxo do Kinesis Data Firehose, um tópico do Amazon MSK, um bucket do Amazon S3 e assim por diante. Para ter mais informações, consulte Coletores de API de tabela.

Depois de criar, compilar e empacotar o código da aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon S3. Em seguida, crie um aplicativo do Managed Service for Apache Flink. Insira a localização do pacote do código, um tópico do Amazon MSK como fonte de dados de transmissão e, normalmente, um local de transmissão ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos

Antes de iniciar este tutorial, conclua duas primeiras etapas de Introdução ao Amazon Managed Service para Apache Flink (DataStream API):

- Etapa 1: Configurar uma conta da AWS e criar um usuário administrador
- Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Para começar, consulte o Criar um aplicativo.

Criar e executar um aplicativo Managed Service for Apache Flink

Neste exercício, você cria um aplicativo Managed Service for Apache Flink com um tópico do Amazon MSK como origem e um bucket do Amazon S3 como coletor.

Esta seção contém as seguintes etapas.

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixar e examinar o código Java Apache Flink Streaming
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Próxima etapa

Criar recursos dependentes

Antes de criar um Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Uma nuvem privada virtual (VPC) com base na Amazon VPC e em um cluster do Amazon MSK
- Um bucket do Amazon S3 para armazenar o código e saída do aplicativo (`ka-app-code-<username>`)

Criar uma VPC e um cluster do Amazon MSK

Para criar uma VPC e cluster do Amazon MSK para acessar a partir do seu aplicativo Managed Service for Apache Flink, siga o tutorial Conceitos básicos do uso do Amazon MSK.

Ao concluir o tutorial, observe o seguinte:

- Registre a lista de servidores bootstrap do seu cluster. Você pode obter a lista de servidores bootstrap com o seguinte comando, substituindo *ClusterArn* pelo nome do recurso da Amazon (ARN) do seu cluster 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"
}
```

- Ao seguir as etapas dos tutoriais, certifique-se de usar a região AWS selecionada no código, nos comandos e nas entradas do console.

Criar um bucket do Amazon S3

Você pode criar um bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esse recurso, consulte os tópicos a seguir:

- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **ka-app-code-*<username>***.

Outros recursos

Quando você cria seu aplicativo, o Managed Service for Apache Flink cria os seguintes CloudWatch recursos da Amazon, caso eles ainda não existam:

- Um grupo de logs chamado `/AWS/KinesisAnalytics-java/MyApplication`.
- Um fluxo de logs chamado `kinesis-analytics-log-stream`.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostras no tópico do Amazon MSK para processamento pelo aplicativo.

1. Conecte-se à instância cliente que você criou na Etapa 4: Criar uma máquina cliente do tutorial Conceitos básicos do uso do Amazon MSK.
2. Instale o Python3, o Pip e a biblioteca Kafka Python:

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

3. Crie um arquivo denominado `stock.py` com o seguinte conteúdo: Substitua o valor `BROKERS` pela lista de agentes de bootstrap que você registrou 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. Mais adiante neste tutorial, você executa o script `stock.py` para enviar dados para o aplicativo.

```
$ python3 stock.py
```

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub.

Para baixar o código de aplicativo Java

1. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/GettingStartedTable`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo Project Object Model (pom.xml) contém informações sobre a configuração e as dependências da aplicação, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `StreamingJob.java` contém o método `main` que define a funcionalidade do aplicativo.
- O aplicativo usa um `FlinkKafkaConsumer` para ler o tópico do Amazon MSK. O trecho a seguir cria um objeto `FlinkKafkaConsumer`:

```
final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProps);
```

- Seu aplicativo cria conectores de origem e de destino para acessar recursos externos usando os objetos `StreamExecutionEnvironment` e `TableEnvironment`.

- O aplicativo cria conectores de origem e coletor usando propriedades dinâmicas do aplicativo, para que você possa especificar os parâmetros do aplicativo (como o bucket do S3) sem recompilar o 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 obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

> **Note**
>
> Ao criar seu aplicativo, é altamente recomendável criar e executar o aplicativo Managed Service for Apache Flink na mesma região do cluster do Amazon MSK. Isso ocorre porque o conector Flink Kafka é, por padrão, otimizado para ambientes de baixa latência. Se você precisar consumir de um cluster Kafka entre regiões, considere aumentar o valor da configuração para `receive.buffer.byte`, como 2097152.
> Para mais informações, consulte Configurações padrão do MSK.

Compilar o código do aplicativo

Nesta seção, você usa o compilador do Apache Maven para criar o código Java para o aplicativo. Para obter informações sobre como instalar o Apache Maven e o Java Development Kit (JDK), consulte Pré-requisitos para concluir os exercícios.

Para compilar o código do aplicativo

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Há duas formas de compilar e empacotar o código:

   - Use a ferramenta de linha de comando do Maven. Crie seu arquivo JAR executando o seguinte comando no diretório que contém o arquivo `pom.xml`:

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

   - Use o ambiente de desenvolvimento. Consulte a documentação de seu ambiente de desenvolvimento para obter mais detalhes.

     > **Note**
     >
     > O código-fonte fornecido depende de bibliotecas do Java 11.

   Você pode carregar o pacote como um arquivo JAR, ou pode compactar o pacote e carregá-lo como um arquivo ZIP. Se você criar seu aplicativo usando a AWS CLI, especifique o tipo de conteúdo de código (JAR ou ZIP).

2. Se houver erros durante a compilação, verifique se sua variável de ambiente `JAVA_HOME` está definida corretamente.

Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

faça o upload do código Java Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon S3 e faz upload do código do seu aplicativo.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira **ka-app-code-*<username>*** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).
5. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
6. Selecione Criar bucket.
7. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior. Selecione Next (Próximo).
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.

- Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).

4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Editar a política do IAM

Edite a política do IAM para adicionar permissões para acessar o bucket do Amazon S3.

Editar a política do IAM para adicionar permissões do bucket do S3

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à 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 o aplicativo

Siga o procedimento a seguir para configurar o aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Criar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
FlinkApplicationProperties	**kafka-topic**	**AWSKafkaTutorialTopic**
FlinkApplicationProperties	**brokers**	***Your Amazon MSK cluster's Bootstrap Brokers list***
FlinkApplicationProperties	**s3Path**	**ka-app-code- *<username>***
FlinkApplicationProperties	**security.protocol**	**SSL**
FlinkApplicationProperties	**ssl.truststore.location**	**/usr/lib/jvm/java-11-amazon-corretto**

ID do grupo	Chave	Valor
		/lib/security/cacerts
FlinkApplicationProperties	**ssl.truststore.password**	**changeit**

6. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
7. Para CloudWatch registrar, marque a caixa de seleção Ativar.
8. Na seção Nuvem privada virtual (VPC), selecione Configuração da VPC com base no cluster do Amazon MSK. Escolha AWSKafkaTutorialCluster.
9. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

Siga o procedimento a seguir para executar o aplicativo.

Executar o aplicativo

1. Na MyApplicationpágina, escolha Executar. Confirme a ação.
2. Quando o aplicativo estiver em execução, atualize a página. O console mostra o Gráfico do aplicativo.
3. Em seu cliente Amazon EC2, execute o script Python que você criou anteriormente para gravar registros no cluster do Amazon MSK para que seu aplicativo processe:

```
$ python3 stock.py
```

Interromper o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Próxima etapa

Limpar recursos AWS

Limpar recursos AWS

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Introdução (API de tabela).

Este tópico contém as seguintes seções.

- Excluir o seu aplicativo Managed Service for Apache Flink
- Excluir seu cluster do Amazon MSK
- Excluir sua VPC
- Excluir seus objetos e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos
- Próxima etapa

Excluir o seu aplicativo Managed Service for Apache Flink

Siga o procedimento a seguir para excluir o aplicativo.

Para excluir o aplicativo

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seu cluster do Amazon MSK

Para excluir seu cluster do Amazon MSK, siga a Etapa 8: Excluir o cluster do Amazon MSK no Guia do desenvolvedor Amazon Managed Streaming for Apache Kafka.

Excluir sua VPC

Para excluir sua Amazon VPC, faça o seguinte:

- Abra o console da Amazon VPC.
- Selecione sua VPC.
- Em Actions (Ações), escolha Delete VPC (Excluir a VPC).

Excluir seus objetos e bucket do Amazon S3

Siga o procedimento a seguir para excluir objetos e bucket do S3.

Excluir seus objetos e bucket do S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o <username>balde ka-app-code-.
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

Siga o procedimento a seguir para excluir seus recursos do IAM.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.

8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

Use o procedimento a seguir para excluir seus CloudWatch recursos.

Para excluir seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Próxima etapa

Próximos Passos

Próximos Passos

Agora que você criou e executou um aplicativo Managed Service for Apache Flink que usa a API de tabela, veja Etapa 5: próximas etapas no Introdução ao Amazon Managed Service para Apache Flink (DataStream API).

Conceitos básicos do Amazon Managed Service for Apache Flink para Python

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink usando Python e a API de tabela. Ela descreve as opções disponíveis para criar e testar seus aplicativos. Ela também fornece instruções para instalar as ferramentas necessárias para concluir os tutoriais deste guia e criar seu primeiro aplicativo.

Tópicos

> **Note**
>
> Se você estiver desenvolvendo o aplicativo Python Flink em um novo Mac com o chip Apple Silicon, poderá encontrar alguns problemas conhecidos com as dependências do Python da versão 1.15. PyFlink Nesse caso, recomendamos executar o interpretador Python no Docker. Para step-by-step obter instruções, consulte Desenvolvimento da PyFlink versão 1.15 no Apple Silicon Mac.

Conceitos básicos do Pyflink - O intérprete de Python para Apache | Amazon Web Services

Antes de começar, recomendamos que você assista ao vídeo a seguir:

Conceitos básicos do Pyflink - O intérprete de Python para Apache | Amazon Web Services

Componentes do aplicativo do Managed Service for Apache Flink

Para processar dados, seu aplicativo do Managed Service for Apache Flink usa um aplicativo Python que processa a entrada e produz a saída usando o runtime do Apache Flink.

O aplicativo do Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.
- Fonte da tabela: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um tópico do Amazon MSK ou similar. Para ter mais informações, consulte Fontes da API de tabela.
- Funções: o aplicativo processa dados usando uma ou mais funções. Uma função pode transformar, enriquecer ou agregar dados.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis, um fluxo do Kinesis Data Firehose, um tópico do Amazon MSK, um bucket do Amazon S3 e assim por diante. Para ter mais informações, consulte Coletores de API de tabela.

Depois de criar o código do seu aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon S3. Em seguida, crie um aplicativo do Managed Service for Apache Flink. Insira a localização do pacote do código, uma fonte de dados de transmissão e, normalmente, um local de transmissão ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos

Antes de iniciar este tutorial, conclua duas primeiras etapas de Introdução ao Amazon Managed Service para Apache Flink (DataStream API):

- Etapa 1: Configurar uma conta da AWS e criar um usuário administrador
- Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Para começar, consulte o Criar um aplicativo.

Criar e executar um aplicativo Managed Service for Apache Flink para o aplicativo Python

Neste exercício, você cria um aplicativo Managed Service for Apache Flink para aplicativo Python com uma transmissão Kinesis como fonte e coletor.

Esta seção contém as seguintes etapas.

Criar recursos dependentes

Antes de criar um Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Duas transmissões do Kinesis para entrada e saída.
- Um bucket do Amazon S3 para armazenar o código e saída do aplicativo (`ka-app-code-<username>`)

Criar duas transmissões do Kinesis

Antes de criar um aplicativo do Managed Service for Apache Flink para este exercício, crie dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`). O aplicativo usa esses fluxos para os fluxos de origem e de destino do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou o comando da AWS CLI a seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro fluxo (`ExampleInputStream`), use o comando `create-stream` AWS CLI do Amazon Kinesis a seguir.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Para criar o segundo fluxo que o aplicativo usa para gravar a saída, execute o mesmo comando, alterando o nome da transmissão para `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Criar um bucket do Amazon S3

Você pode criar um bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esse recurso, consulte os tópicos a seguir:

- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Outros recursos

Quando você cria seu aplicativo, o Managed Service for Apache Flink cria os seguintes CloudWatch recursos da Amazon, caso eles ainda não existam:

- Um grupo de logs chamado `/AWS/KinesisAnalytics-java/MyApplication`.
- Um fluxo de logs chamado `kinesis-analytics-log-stream`.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

> **Note**
>
> O script do Python nesta seção usa o AWS CLI. Você deve configurar seu AWS CLI para usar as credenciais da sua conta e a região padrão. Para configurar o seu AWS CLI, digite o seguinte:
>
> ```
> aws configure
> ```

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o restante do tutorial.

Criar e examinar o código Python de transmissão do Apache Flink

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/python/GettingStarted`.

O código do aplicativo está localizado no arquivo `getting_started.py`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem de tabela do Kinesis para ler o fluxo de origem. O trecho a seguir chama a função `create_table` para criar a origem de tabela Kinesis:

```
table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region)
```

A função `create_table` usa um comando SQL para criar uma tabela que é apoiada pela origem de transmissão:

```
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)
 }
```

- O aplicativo cria duas tabelas e, em seguida, grava o conteúdo de uma tabela na outra.

```
    # 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))
```

- O aplicativo usa o conector Flink, do arquivo flink- sql-connector-kinesis _2.12/1.15.2.

Adicionar dependências de terceiros aos aplicativos Python

Ao usar pacotes Python de terceiros (como boto3), você precisará adicionar suas dependências transitivas e as propriedades necessárias para direcionar a essas dependências. Em um alto nível, para PyPi dependências, você pode copiar os arquivos e pastas que estão localizados na `site-packages` pasta de ambientes python para criar uma estrutura de diretórios como abaixo:

```
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 adicionar o boto3 como uma dependência de terceiros:

1. Crie um ambiente Python autônomo (conda ou similar) em sua máquina local com as dependências necessárias.
2. Observe a lista inicial de pacotes na pasta `site_packages` desse ambiente.
3. `pip-install` todas as dependências necessárias para seu aplicativo.
4. Observe os pacotes que foram adicionados à pasta `site_packages` após a etapa 3 acima. Essas são as pastas que você precisa incluir em seu pacote (abaixo da pasta `my_deps`), organizadas conforme mostrado acima. Isso permitirá que você capture uma diferença dos

pacotes entre as etapas 2 e 3 para identificar as dependências de pacotes corretas para seu aplicativo.

5. Forneça `my_deps/` como argumento para a propriedade `pyFiles` no grupo de propriedades `kinesis.analytics.flink.run.options`, conforme descrito abaixo para a propriedade `jarfiles`. O Flink também permite que você especifique dependências do Python usando função add_python_file, mas é importante ter em mente que você só precisa especificar uma ou outra, não ambas.

> **Note**
>
> Você não precisa nomear a pasta `my_deps`. A parte importante é registrar as dependências usando `pyFiles` ou `add_python_file`. Um exemplo pode ser encontrado em Como usar o boto3 no PyFlink.

Fazer upload do código de transmissão Python do Apache Flink

Nesta seção, você cria um bucket do Amazon S3 e faz upload do código do seu aplicativo.

Para fazer o upload do código do aplicativo usando o console:

1. Use seu aplicativo de compactação preferido para compactar os arquivos `getting-started.py` e https://mvnrepository.com/artifact/org.apache.flink/flink - sql-connector-kinesis_2.12/1.15.2. Nomeie o arquivo como `myapp.zip`. Se você incluir a pasta externa em seu arquivo, você deve inclui-la no caminho com o código em seu(s) arquivo(s) de configuração: `GettingStarted/getting-started.py`.
2. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
3. Selecione Criar bucket.
4. Insira **`ka-app-code-<username>`** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
5. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).
6. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
7. Selecione Criar bucket.

8. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
9. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `myapp.zip` que você criou na etapa anterior. Selecione Next (Próximo).
10. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

Para fazer o upload do código do aplicativo usando o AWS CLI:

> **Note**
>
> Não use os recursos de compactação no Finder (macOS) ou no Windows Explorer (Windows) para criar o arquivo `myapp.zip`. Isso pode resultar em um código de aplicativo inválido.

1. Use seu aplicativo de compactação preferido para compactar os arquivos `streaming-file-sink.py` e https://mvnrepository.com/artifact/org.apache.flink/flink - sql-connector-kinesis _2.12/1.15.2.

   > **Note**
   >
   > Não use os recursos de compactação no Finder (macOS) ou no Windows Explorer (Windows) para criar o arquivo myapp.zip. Isso pode resultar em um código de aplicativo inválido.

2. Use seu aplicativo de compactação preferido para compactar os arquivos `getting-started.py` e https://mvnrepository.com/artifact/org.apache.flink/ flink-sql-connector-kinesis /1.15.2. Nomeie o arquivo como `myapp.zip`. Se você incluir a pasta externa em seu arquivo, você deve inclui-la no caminho com o código em seu(s) arquivo(s) de configuração: `GettingStarted/getting-started.py`.
3. Execute o seguinte comando:

   ```
   $ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
   ```

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pela aplicação.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

Siga o procedimento a seguir para configurar o aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.

2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **myapp.zip**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

   Selecione Save (Salvar).
6. Em Propriedades, selecione Adicionar grupo novamente.
7. Insira o seguinte:

ID do grupo	Chave	Valor
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Em Propriedades, selecione Adicionar grupo novamente. Em ID do grupo, digite **kinesis.analytics.flink.run.options**. Esse grupo de propriedades especiais informa ao aplicativo onde encontrar seus recursos de código. Para ter mais informações, consulte Especificar seus arquivos de código.
9. Insira o seguinte:

ID do grupo	Chave	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. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
11. Para CloudWatch registrar, escolha a caixa de seleção Ativar.
12. Escolha Atualizar.

> **Note**
>
> Quando você opta por habilitar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Editar a política do IAM

Edite a política do IAM para adicionar permissões para acessar o bucket do Amazon S3.

Editar a política do IAM para adicionar permissões do bucket do S3

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Próxima etapa

Limpar recursos AWS

Limpar recursos AWS

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial de conceitos básicos (Python).

Este tópico contém as seguintes seções.

Excluir o seu aplicativo Managed Service for Apache Flink

Siga o procedimento a seguir para excluir o aplicativo.

Para excluir o aplicativo

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seus fluxos de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e, em seguida, confirme a exclusão.

Excluir seus objetos e bucket do Amazon S3

Siga o procedimento a seguir para excluir objetos e bucket do S3.

Excluir seus objetos e bucket do S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o <username>balde ka-app-code-.
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

Siga o procedimento a seguir para excluir seus recursos do IAM.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

Use o procedimento a seguir para excluir seus CloudWatch recursos.

Para excluir seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e confirme a exclusão.

Conceitos básicos (Scala)

> **Note**
>
> A partir da versão 1.15, o Flink não tem mais o Scala. Agora, os aplicativos podem usar a API Java de qualquer versão do Scala. O Flink ainda usa o Scala em alguns componentes importantes internamente, mas não expõe o Scala no carregador de classes do código do usuário. Por isso, os usuários precisam adicionar dependências do Scala em seus arquivos jar.
>
> Para obter mais informações sobre as mudanças do Scala no Flink 1.15, consulte Sem o Scala na versão 1.15.

Neste exercício, você cria um aplicativo do Managed Service for Apache Flink para o Scala com fluxos do Kinesi como fonte e coletor.

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Compile e faça o upload do código do aplicativo
- Criar e executar o aplicativo (console)
- Crie e execute o aplicativo (CLI)
- Limpar AWSrecursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Duas transmissões do Kinesis para entrada e saída.
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos. consulte os tópicos a seguir:

- Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seus fluxos de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.

  Como criar os fluxos de dados (AWS CLI)

  - Para criar o primeiro fluxo (`ExampleInputStream`), use o seguinte comando AWS CLI do Amazon Kinesis.

    ```
    aws kinesis create-stream \
        --stream-name ExampleInputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

  - Execute o mesmo comando, alterando o nome do fluxo para `ExampleOutputStream`, a fim de criar o segundo fluxo que o aplicativo usará para gravar a saída.

    ```
    aws kinesis create-stream \
        --stream-name ExampleOutputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-`*`<username>`***.

Outros recursos

Quando você cria seu aplicativo, o Managed Service for Apache Flink cria os seguintes recursos do Amazon CloudWatch, caso eles ainda não existam:

- Um grupo de logs chamado `/AWS/KinesisAnalytics-java/MyApplication`
- Um fluxo de logs chamado `kinesis-analytics-log-stream`

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

> **Note**
>
> O script do Python nesta seção usa o AWS CLI. Você deve configurar seu AWS CLI para usar as credenciais da sua conta e a região padrão. Para configurar o seu AWS CLI, digite o seguinte:
>
> ```
> aws configure
> ```

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código Python do aplicativo desta amostra está disponível no GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo `build.sbt` contém informações sobre a configuração e as dependências do aplicativo, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.scala` contém o método principal que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do 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)
}
```

O aplicativo também usa um coletor do Kinesis para gravar no fluxo de resultados. O trecho a seguir cria o coletor do 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
}
```

- O aplicativo cria conectores de origem e de destino para acessar recursos externos usando um objeto StreamExecutionEnvironment.
- O aplicativo cria conectores de origem e de coletores usando propriedades dinâmicas do aplicativo. As propriedades de runtime do aplicativo para ler e configurar os conectores. Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compile e faça o upload do código do aplicativo

Nesta seção, você compila e carrega o código do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

Compilar o código do aplicativo

Nesta seção, você usa a ferramenta de compilação SBT para criar o código do Scala para o aplicativo. Para instalar o SBT, consulte Instalar o sbt com a configuração cs. Você também precisa instalar o Java Development Kit (JDK). Consulte Pré-requisitos para concluir os exercícios.

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Você pode compilar e empacotar seu código com o SBT:

```
sbt assembly
```

2. Se o aplicativo é compilado com êxito, o arquivo a seguir é criado:

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

Faça o upload do código Scala do Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon S3 e faz upload do código do seu aplicativo.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira `ka-app-code-<username>` no campo Bucket name (Nome do bucket). Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Em Configurar opções, mantenha as configurações como estão e selecione Próximo.
5. Em Definir permissões, mantenha as configurações como estão e selecione Próximo.
6. Selecione Create bucket (Criar bucket).
7. Selecione o bucket `ka-app-code-<username>` e selecione Fazer upload.
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `getting-started-scala-1.0.jar` que você criou na etapa anterior.
9. Você não precisa alterar nenhuma das configurações para o objeto, então selecione Fazer upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo do

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Application name (Nome do aplicativo), insira **`MyApplication`**.
   - Em Descrição, insira **`My scala test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de criar um perfil do IAM e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do seu aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

Siga o procedimento a seguir para configurar o aplicativo.

Configurar o aplicativo

1. Na página MyApplication, selecione Configure (Configurar).

2. Na página Configure application (Configurar aplicativo), forneça o Code location (Local do código):
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **getting-started-scala-1.0.jar.**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Selecione Save (Salvar).
6. Em Propriedades, selecione Adicionar grupo novamente.
7. Insira o seguinte:

ID do grupo	Chave	Valor
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Em Monitoring (Monitoramento), confirme se Monitoring metrics level (Nível de monitoramento de métricas) está definido como Application (Aplicativo).
9. Em Registro em log do CloudWatch, marque a caixa de seleção Habilitar.

10. Selecione Atualizar.

> **Note**
>
> Quando você opta por habilitar o Amazon CloudWatch Logs, o Managed Service for Apache Flink cria um grupo de logs e um fluxo de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Editar a política do IAM

Edite a política do IAM para adicionar permissões para acessar o bucket do Amazon S3.

Editar a política do IAM para adicionar permissões do bucket do S3

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Para interromper o aplicativo, na página Meu aplicativo, selecione Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa a AWS Command Line Interface para criar e executar o aplicativo Flink do Managed Service for Apache Flink. Use o comando AWS CLI kinesisanalyticsv2 para criar e interagir com os aplicativos Managed Service para aplicativos Apache Flink.

Criação de uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação de ler no fluxo de origem, e outra que concede permissões para ações de gravação no fluxo de coleta. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua **`username`** pelo nome de usuário que você

usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos nomes do recurso da Amazon Resources Names (ARNs) **(012345678901)**pelo ID da sua conta.

```
{
    "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 obter instruções passo a passo para criar uma política de permissões, consulte Tutorial: crie e anexe a sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

Crie uma política do IAM.

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma função do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.

2. No painel de navegação, selecione Perfis e Criar perfil.
3. Em Selecionar tipo de identidade de confiança, selecione Serviço AWS.
4. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis.
5. Em Selecione seu caso de uso, selecione Managed Service for Apache Flink.
6. Selecione Next: Permissions (Próximo: permissões).
7. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
8. Na página Create role (Criar função), insira **`MF-stream-rw-role`** em Role name (Nome da função). Selecione Create role (Criar função).

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
9. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, Crie uma política de permissões.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).
   b. Selecione Attach Policies.
   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).
   d. Selecione a política `AKReadSourceStreamWriteSinkStream` e selecione Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para obter instruções passo a passo sobre como criar um perfil, consulte Criação de uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo

Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (nome do usuário) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (012345678901) na função de execução do serviço pelo seu ID da conta.

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

Execute o CreateApplication com a seguinte solicitação para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação StartApplication para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode analisar as métricas do Managed Service for Apache Flink no console do Amazon CloudWatch para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação StopApplication para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Execute a ação `StopApplication` com a solicitação anterior para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de registro em log do CloudWatch

Você pode usar o AWS CLI para adicionar um fluxo do Amazon CloudWatch Log ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consulte Configurar o log de aplicativos.

Atualizar propriedades do ambiente

Nesta seção, você usa a ação UpdateApplication para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando você precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação CLI UpdateApplication.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da ação `UpdateApplication` a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do

aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `ListApplications` ou `DescribeApplication`. Atualize o sufixo do nome do bucket (<username>) com o sufixo que você selecionou na seção Criar recursos dependentes.

```
{{
    "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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Limpar AWSrecursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janela em cascata.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Excluir seus recursos do CloudWatch

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel do Managed Service for Apache Flink, selecione MyApplication.
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, selecione ExampleInputStream.
3. Na página ExampleInputStream, selecione Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, selecione ExampleOutputStream, selecione Ações, selecione Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione o bucket ka-app-code-***<username>***.
3. Selecione Excluir e insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Selecione a política kinesis-analytics-service-MyApplication-us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Selecione a função Kinesis-Analytics-MyApplication-US-West-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Excluir seus recursos do CloudWatch

1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. Na barra de navegação, selecione Logs.
3. Selecione o grupo de logs /aws/kinesis-analytics/MyApplication.
4. Selecione Excluir grupo de logs e confirme a exclusão.

Criar aplicativos Managed Service for Apache Flink com o Apache Beam

Você pode usar a estrutura Apache Beam com seu aplicativo Managed Service for Apache Flink para processar dados de streaming. Os aplicativos Managed for Apache Flink que usam o Apache Beam usam o Apache Flink runner para executar pipelines do Beam.

Para obter um tutorial sobre como usar o Apache Beam em um aplicativo Managed Service for Apache Flink, consulte. Usar CloudFormation for Managed Service for Apache Flink

Este tópico contém as seguintes seções:

- Usar o Apache Beam com Managed Service for Apache Flink
- Capacidades do Beam
- Criar um aplicativo usando o Apache Beam

Usar o Apache Beam com Managed Service for Apache Flink

Observe o seguinte sobre o uso do Apache Flink Runner com o Managed Service para Apache Flink:

- As métricas do Apache Beam não podem ser visualizadas no console Managed Service for Apache Flink.
- O Apache Beam só é compatível com aplicativos Managed Service for Apache Flink que usam o Apache Flink versão 1.8 e superior. O Apache Beam só é compatível com aplicativos Managed Service for Apache Flink que usam o Apache Flink versão 1.6.

Capacidades do Beam

O Managed Service for Apache Flink suporta os mesmos recursos do Apache Beam que o executor Apache Flink. Para obter informações sobre quais atributos são compatíveis com o executor Apache Flink, consulte a Matriz de compatibilidade do Beam.

Recomendamos que você teste seu aplicativo Apache Flink no serviço Managed Service for Apache Flink para verificar se oferecemos suporte a todos os recursos de que seu aplicativo precisa.

Criar um aplicativo usando o Apache Beam

Neste exercício, você cria um aplicativo Managed Service for Apache Flink que transforma dados usando o Apache Beam. O Apache Beam é um modelo de programação para processar dados de streaming. Para obter informações sobre como usar o Apache Beam com o Managed Service para Apache Flink, consulte. Usar o Apache Beam

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício. Introdução (DataStream API)

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Compilar o código do aplicativo
- Fazer upload do código Java Apache Flink Streaming
- Crie e execute o aplicativo Managed Service for Apache Flink
- Limpar AWS recursos
- Próximas etapas

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos. consulte os tópicos a seguir:

- Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seus fluxos de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar strings aleatórias no stream para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `ping.py` com o conteúdo a seguir.

```
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. Execute o script `ping.py`:

```
$ python ping.py
```

   Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código de aplicativo Java deste exemplo está disponível no GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/Beam`.

O código do aplicativo está localizado no arquivo `BasicBeamStreamingJob.java`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa o Apache Beam ParDo para processar registros recebidos invocando uma função de transformação personalizada chamada `PingPongFn`.

  O código para invocar a função `PingPongFn` é o seguinte:

```
.apply("Pong transform",
    ParDo.of(new PingPongFn())
```

- O serviço gerenciado para aplicativos Apache Flink que usam o Apache Beam requer os seguintes componentes. Se você não incluir esses componentes e versões no seu `pom.xml`, seu aplicativo carregará as versões incorretas das dependências do ambiente e, como as versões não coincidem, seu aplicativo falhará no runtime.

```
<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>
```

- A função de transformação `PingPongFn` passa os dados de entrada para o fluxo de saída, a menos que os dados de entrada sejam ping. Nesse caso, ela emite a string pong\npara o fluxo de saída.

O código da função de transformação é o seguinte:

```
    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 o código do aplicativo

Para compilar o aplicativo, faça o seguinte:

1. Instale o Java e o Maven, caso ainda não o tenha feito. Para obter mais informações, consulte Pré-requisitos no tutorial Introdução (DataStream API).
2. Compile o aplicativo com o seguinte comando:

```
mvn package -Dflink.version=1.15.3 -Dflink.version.minor=1.8
```

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

A compilação do aplicativo cria o arquivo JAR do aplicativo (`target/basic-beam-app-1.0.jar`).

Fazer upload do código Java Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. No console do Amazon S3, escolha o bucket ka-app-code-***<nomedousuário>*** e escolha Fazer upload.
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `basic-beam-app-1.0.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Crie e execute o aplicativo Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo do

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Application name (Nome do aplicativo), insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, escolha Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Escolha Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de criar um perfil do IAM e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesis-analytics-MyApplication-us-west-2`

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Escolha Policies (Políticas). Escolha a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), escolha Edit policy (Editar política). Escolha a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na página MyApplication, escolha Configure (Configurar).

2. Na página Configure application (Configurar aplicativo), forneça o Code location (Local do código):
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **basic-beam-app-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, escolha Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Insira o seguinte:

ID do grupo	Chave	Valor
BeamApplicationProperties	**InputStreamName**	**ExampleInputStream**
BeamApplicationProperties	**OutputStreamName**	**ExampleOutputStream**
BeamApplicationProperties	**AwsRegion**	**us-west-2**

5. Em Monitoring (Monitoramento), confirme se Monitoring metrics level (Nível de monitoramento de métricas) está definido como Application (Aplicativo).
6. Em CloudWatch logging (Registro em log do CloudWatch), marque a caixa de seleção Enable (Habilitar).
7. Escolha Atualizar.

> **Note**
>
> Quando você opta por habilitar o registro em log do CloudWatch, o Managed Service for Apache Flink cria um grupo de logs e um fluxo de logs. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Você pode verificar as métricas do Managed Service for Apache Flink no console do CloudWatch para confirmar que o aplicativo está funcionando.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janela em cascata.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Excluir seus recursos do CloudWatch

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel do Managed Service for Apache Flink, selecione MyApplication.
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, selecione ExampleInputStream.
3. Na página ExampleInputStream, selecione Excluir Kinesis Stream e, em seguida, confirme a exclusão.

4. Na páginaKinesis Streams, selecione ExampleOutputStream, selecione Ações, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione o bucket ka-app-code-***<username>***.
3. Escolha Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Selecione a política kinesis-analytics-service-MyApplication-us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Selecione a função Kinesis-Analytics-MyApplication-US-West-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Excluir seus recursos do CloudWatch

1. Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. Na barra de navegação, selecione Logs.
3. Selecione o grupo de logs /aws/kinesis-analytics/MyApplication.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Próximas etapas

Agora que você criou e executou um aplicativo básico do Managed Service for Apache Flink que transforma dados usando o Apache Beam, consulte o aplicativo a seguir para obter um exemplo de uma solução mais avançada do Managed Service for Apache Flink.

- Workshop sobre o Beam on Managed Service for Apache Flink Streaming: Neste workshop, exploramos um exemplo completo que combina aspectos de lote e streaming em um pipeline uniforme do Apache Beam.

Workshops de treinamento, laboratórios e implementações de soluções

Os end-to-end exemplos a seguir demonstram soluções avançadas de serviços gerenciados para Apache Flink.

Tópicos

- Desenvolver aplicativos Apache Flink localmente antes de implantar no Managed Service for Apache Flink for Apache Flink
- Detecção de eventos com o Managed Service for Apache Flink Studio
- AWS Solução de dados de transmissão para o Amazon Kinesis
- Clickstream Lab com Apache Flink e Apache Kafka
- Escalabilidade personalizada usando o ajuste de escala automático do aplicativo
- CloudWatch Painel da Amazon
- AWS Solução de dados de transmissão para o Amazon MSK
- Mais serviços gerenciados para soluções Apache Flink em GitHub

Desenvolver aplicativos Apache Flink localmente antes de implantar no Managed Service for Apache Flink for Apache Flink

Este workshop mostrará as noções básicas sobre como começar a desenvolver aplicativos Apache Flink localmente com o objetivo de longo prazo de implantar no Managed Service for Apache Flink for Apache Flink.

A solução pode ser encontrada aqui: Guia para iniciantes sobre desenvolvimento local com o Apache Flink

Detecção de eventos com o Managed Service for Apache Flink Studio

Este workshop descreve a detecção de eventos com o Managed Service for Apache Flink Studio e sua implantação como um aplicativo Managed Service for Apache Flink

A solução pode ser encontrada aqui: Detecção de eventos com o Managed Service for Apache Flink for Apache Flink

AWS Solução de dados de transmissão para o Amazon Kinesis

A solução de dados de transmissão para o Amazon Kinesis da AWS configura automaticamente os serviços da AWS necessários para capturar, armazenar, processar e entregar dados de transmissão com facilidade. A solução oferece várias opções para resolver casos de uso de dados de transmissão. A opção Managed Service for Apache Flink fornece um exemplo de ETL de end-to-end streaming que demonstra um aplicativo do mundo real que executa operações analíticas em dados simulados de táxi de Nova York.

Cada solução inclui os seguintes componentes:

- Um pacote do AWS CloudFormation para implantar todo o exemplo.
- Um CloudWatch painel para exibir as métricas do aplicativo.
- CloudWatch alarmes sobre as métricas de aplicação mais relevantes.
- Todos os perfis do IAM e políticas necessários.

A solução pode ser encontrada em Solução de dados de transmissão para o Amazon Kinesis

Clickstream Lab com Apache Flink e Apache Kafka

Um laboratório completo para casos de uso de clickstream usando Amazon Managed Streaming for Apache Kafka para armazenamento de transmissão e aplicativos Managed Service for Apache Flink for Apache Flink para processamento de fluxos.

A solução pode ser encontrada aqui: Clickstream Lab

Escalabilidade personalizada usando o ajuste de escala automático do aplicativo

Uma amostra que ajuda os usuários a escalar automaticamente o Managed Service para aplicativos Apache Flink usando o Application Auto Scaling. Isso permite que os usuários configurem políticas e atributos de escalabilidade personalizados.

As soluções podem ser encontradas aqui:

- Serviço gerenciado para escalonamento automático do aplicativo Apache Flink
- Escalabilidade programada

Para obter mais informações sobre como você pode realizar escalabilidade personalizada, consulte Habilitar escalabilidade programada e baseada em métricas para o Amazon Managed Service para Apache Flink.

CloudWatch Painel da Amazon

Um exemplo de CloudWatch painel para monitorar o Managed Service para aplicativos Apache Flink. O painel de exemplo também inclui um aplicativo de demonstração para ajudar a demonstrar a funcionalidade do painel.

A solução pode ser encontrada aqui: Painel de métricas do Managed Service for Apache Flink

AWS Solução de dados de transmissão para o Amazon MSK

A solução de dados de transmissão AWS para o Amazon MSK fornece modelos do AWS CloudFormation em que os dados fluem por produtores, armazenamento de transmissão, consumidores e destinos.

A solução pode ser encontrada em AWSSolução de dados de transmissão para o Amazon MSK

Mais serviços gerenciados para soluções Apache Flink em GitHub

Os end-to-end exemplos a seguir demonstram soluções avançadas de serviços gerenciados para Apache Flink e estão disponíveis em: GitHub

- Amazon Managed Service for Apache Flink Flink – Utilitário de avaliação comparativa
- Gerenciador de instantâneos – Amazon Managed Service for Apache Flink for Apache Flink
- Streaming ETL com Apache Flink e Amazon Managed Service for Apache Flink
- Análise de sentimentos em tempo real com base no feedback do cliente

Utilitários

Os utilitários a seguir podem facilitar o uso do serviço Managed Service for Apache Flink:

Tópicos

Gerenciador de snapshots

É uma prática recomendada que os aplicativos Flink acionem regularmente pontos de salvamento/snapshots para permitir uma recuperação de falhas mais perfeita. O Gerenciador de snapshots automatiza essa tarefa e oferece os seguintes benefícios:

- tira um novo snapshot de um aplicativo Managed Service for Apache Flink for Apache Flink;
- obtém uma contagem de snapshots do aplicativo;
- verifica se a contagem é maior do que o número necessário de snapshots;
- exclui snapshots mais antigos que são mais antigos do que o número necessário.

Para ver um exemplo, consulte Gerenciador de instantâneos.

Avaliação comparativa

O utilitário de avaliação comparativa Managed Service for Apache Flink ajuda no planejamento de capacidade, no teste de integração e na avaliação comparativa de aplicativos Managed Service for Apache Flink for Apache Flink.

Para ver um exemplo, consulte Avaliação comparativa

Managed Service for Apache Flink: exemplos

Esta seção apresenta exemplos de como criar e trabalhar com aplicativos no Managed Service for Apache Flink. Eles incluem exemplos de código e step-by-step instruções para ajudá-lo a criar serviços gerenciados para aplicativos Apache Flink e testar seus resultados.

Antes de explorar esses exemplos, recomendamos que você analise em primeiro lugar o seguinte :

- Como funcionam
- Introdução (DataStream API)

> **Note**
>
> Esses exemplos pressupõem que você esteja usando a região Oeste dos EUA (Oregon) (`us-west-2`). Se você estiver usando uma região diferente, atualize o código, os comandos e o perfil do IAM do aplicativo de forma adequada.

Tópicos

- DataStream Exemplos de API
- Exemplos de Python
- Exemplo do Scala

DataStream Exemplos de API

Os exemplos a seguir demonstram como criar aplicativos usando a API Apache Flink DataStream .

Tópicos

- Exemplo: janela em cascata
- Exemplo: janela deslizante
- Exemplo: gravando em um bucket do Amazon S3
- Tutorial: Usando um aplicativo Managed Service for Apache Flink para replicar dados de um tópico em um cluster MSK para outro em uma VPC
- Exemplo: Use um consumidor EFO com um fluxo de dados do Kinesis

- Exemplo: Gravando no Kinesis Data Firehose
- Exemplo: Leia a partir de um fluxo de dados do Kinesis em outra conta.
- Tutorial: Usando um Truststore personalizado com o Amazon MSK

Exemplo: janela em cascata

Neste exercício, você cria um aplicativo Managed Service for Apache Flink que agrega dados usando uma janela em cascata. A agregação está habilitada por padrão no Flink. Para desativá-la, use o seguinte:

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Introdução (DataStream API).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seu fluxo de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

   Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/TumblingWindow`.

O código do aplicativo está localizado no arquivo `TumblingWindowStreamingJob.java`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Inclua a seguinte declaração de importação:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- O aplicativo usa o operador `timeWindow` para encontrar a contagem dos valores de cada símbolo de ação em uma janela em cascata de cinco segundos. O código a seguir cria o operador e envia os dados agregados para um novo coletor de fluxo de dados do 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 o código do aplicativo

Para compilar o aplicativo, faça o seguinte:

1. Instale o Java e o Maven, caso ainda não o tenha feito. Para obter mais informações, consulte Pré-requisitos no tutorial Introdução (DataStream API).
2. Compile o aplicativo com o seguinte comando:

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

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

A compilação do aplicativo cria o arquivo JAR do aplicativo (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

faça o upload do código Java Apache Flink Streaming

Nesta seção, você o faz upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.

2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
5. Para CloudWatch registrar, marque a caixa de seleção Ativar.
6. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`
>
> Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Executar o aplicativo

1. Na MyApplicationpágina, escolha Executar. Deixe a opção Executar sem snapshot selecionada e confirme a ação.
2. Quando o aplicativo estiver em execução, atualize a página. O console mostra o Gráfico do aplicativo.

Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console para verificar se o aplicativo está funcionando.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janela em cascata.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.

2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: janela deslizante

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Introdução (DataStream API).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`).
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seus fluxos de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/SlidingWindow`.

O código do aplicativo está localizado no arquivo `SlidingWindowStreamingJobWithParallelism.java`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- O aplicativo usa o operador `timeWindow` para descobrir o valor mínimo para cada símbolo de ação em uma janela de dez segundos que desliza por cinco segundos. O código a seguir cria o operador e envia os dados agregados para um novo coletor de fluxo de dados do Kinesis:
- Inclua a seguinte declaração de importação:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- O aplicativo usa o operador `timeWindow` para encontrar a contagem dos valores de cada símbolo de ação em uma janela em cascata de cinco segundos. O código a seguir cria o operador e envia os dados agregados para um novo coletor de fluxo de dados do 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 o código do aplicativo

Para compilar o aplicativo, faça o seguinte:

1. Instale o Java e o Maven, caso ainda não o tenha feito. Para obter mais informações, consulte Pré-requisitos no tutorial Introdução (DataStream API).
2. Compile o aplicativo com o seguinte comando:

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

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

A compilação do aplicativo cria o arquivo JAR do aplicativo (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

faça o upload do código Java Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket e, em seguida, escolha Upload. <username>
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:

- Em Nome do aplicativo, insira **`MyApplication`**.
- Em Runtime, selecione Apache Flink.
- Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).

4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*`012345678901`*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
5. Para CloudWatch registrar, marque a caixa de seleção Ativar.
6. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`
>
> Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Configure o paralelismo do aplicativo

Este exemplo de aplicativo usa a execução paralela de tarefas. O código do aplicativo a seguir define o paralelismo do operador `min`:

```
.setParallelism(3) // Set parallelism for the min operator
```

O paralelismo do aplicativo não pode ser maior do que o paralelismo provisionado, que tem um padrão de 1. Para aumentar o paralelismo do seu aplicativo, use a seguinte ação AWS CLI:

```
aws kinesisanalyticsv2 update-application
      --application-name MyApplication
      --current-application-version-id <VersionId>
      --application-configuration-update "{\"FlinkApplicationConfigurationUpdate
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
 \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

Você pode recuperar o ID da versão atual do aplicativo usando as ListApplicationsações DescribeApplicationou.

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console para verificar se o aplicativo está funcionando.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janelas deslizantes.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.

2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: gravando em um bucket do Amazon S3

Neste exercício, você cria um Managed Service for Apache Flink que tem um fluxo de dados do Kinesis como origem e um bucket do Amazon S3 como coletor. Usando o coletor, você pode conferir a saída do aplicativo no console do Amazon S3.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Introdução (DataStream API).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Modifique o código do aplicativo
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Verifique a saída do aplicativo
- Opcional: personalize a fonte e o coletor
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Um fluxo de dados do Kinesis (`ExampleInputStream`).

- Um bucket do Amazon S3 para armazenar o código e saída do aplicativo (`ka-app-code-<username>`)

> **Note**
>
> O Managed Service for Apache Flink não pode gravar dados no Amazon S3 com a criptografia do lado do servidor habilitada no Managed Service for Apache Flink.

Você pode criar o fluxo de dados do Kinesis e um bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seu fluxo de dados **`ExampleInputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**. Crie duas pastas (**`code`** e **`data`**) no bucket do Amazon S3.

O aplicativo cria os seguintes CloudWatch recursos, caso eles ainda não existam:

- Um grupo de logs chamado `/AWS/KinesisAnalytics-java/MyApplication`.
- Um fluxo de logs chamado `kinesis-analytics-log-stream`.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
    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. Execute o script `stock.py`:

```
$ python stock.py
```

   Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.

2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/S3Sink`.

O código do aplicativo está localizado no arquivo `S3StreamingSinkJob.java`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Você precisa incluir a seguinte declaração de importação:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- O aplicativo usa um coletor do Apache Flink S3 para gravar no Amazon S3.

  O coletor lê mensagens em uma janela em cascata, codifica mensagens em objetos de bucket do S3 e envia os objetos codificados para o coletor do S3. O código a seguir codifica objetos para envio ao 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**
>
> O aplicativo usa um objeto `StreamingFileSink` Flink para gravar no Amazon S3. Para obter mais informações sobre o`StreamingFileSink`, consulte a StreamingFileSinkdocumentação do Apache Flink.

Modifique o código do aplicativo

Nesta seção, você modifica o código do aplicativo para gravar a saída em seu bucket do Amazon S3.

Atualize a linha a seguir com seu nome de usuário para especificar o local de saída do aplicativo:

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

Compilar o código do aplicativo

Para compilar o aplicativo, faça o seguinte:

1. Instale o Java e o Maven, caso ainda não o tenha feito. Para obter mais informações, consulte Pré-requisitos no tutorial Introdução (DataStream API).
2. Compile o aplicativo com o seguinte comando:

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

A compilação do aplicativo cria o arquivo JAR do aplicativo (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

faça o upload do código Java Apache Flink Streaming

Nesta seção, você o faz upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket, navegue até a pasta de código e escolha Upload. <username>
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Em Nome do aplicativo, insira **`MyApplication`**.

- Em Runtime, selecione Apache Flink.
- Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).

6. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
7. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso ao fluxo de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta. Substitua <username> pelo seu nome de usuário.

```
{
            "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **code/aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
5. Para CloudWatch registrar, marque a caixa de seleção Ativar.
6. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Executar o aplicativo

1. Na MyApplicationpágina, escolha Executar. Deixe a opção Executar sem snapshot selecionada e confirme a ação.
2. Quando o aplicativo estiver em execução, atualize a página. O console mostra o Gráfico do aplicativo.

Verifique a saída do aplicativo

No console do Amazon S3, abra a pasta de dados em seu bucket do S3.

Depois de alguns minutos, os objetos contendo dados agregados do aplicativo serão apresentados.

Note

A agregação está habilitada por padrão no Flink. Para desativá-la, use o seguinte:

```
sink.producer.aggregation-enabled' = 'false'
```

Opcional: personalize a fonte e o coletor

Nesta seção, você personaliza as configurações dos objetos de origem e coletor.

Note

Depois de alterar as seções do código descritas nas seções a seguir, faça o seguinte para recarregar o código do aplicativo:

- Repita as etapas da seção the section called “Compilar o código do aplicativo” para compilar o código atualizado do aplicativo.
- Repita as etapas da seção the section called “faça o upload do código Java Apache Flink Streaming” para fazer o upload do código atualizado do aplicativo.

- Na página do aplicativo no console, selecione Configure e, em seguida, selecione Update (Atualizar) para recarregar o código do aplicativo atualizado em seu aplicativo.

Esta seção contém as seguintes seções:

- Configure o particionamento de dados
- Configure a frequência de leitura
- Configure o buffer de gravação

Configure o particionamento de dados

Nesta seção, você configura os nomes das pastas que o coletor de arquivos de streaming cria no bucket do S3. Para isso, adicione um atribuidor de bucket ao coletor de arquivos de streaming.

Para personalizar os nomes das pastas criados no bucket do S3, faça o seguinte:

1. Adicione as seguintes declarações de importação ao início do arquivo `S3StreamingSinkJob.java`:

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPol
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAss
```

2. Atualize o método `createS3SinkFromStaticConfig()` no código para que fique como se segue:

```
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;
}
```

O exemplo de código anterior usa o `DateTimeBucketAssigner` com um formato de data personalizado para criar pastas no bucket do S3. O `DateTimeBucketAssigner` usa o sistema de horário atual para criar nomes para os buckets. Se você quiser criar um atribuidor de bucket personalizado para personalizar ainda mais os nomes das pastas criadas, você pode criar uma classe que implemente. BucketAssigner Você implementa sua lógica personalizada usando o método `getBucketId`.

Uma implementação personalizada do `BucketAssigner` pode usar o parâmetro Contexto para obter mais informações sobre um registro a fim de determinar sua pasta de destino.

Configure a frequência de leitura

Nesta seção, você configura a frequência das leituras no fluxo de origem.

Por padrão, o consumidor do Kinesis Streams lê o fluxo de origem cinco vezes por segundo. Essa frequência causará problemas se houver mais de um cliente lendo o fluxo ou se o aplicativo precisar tentar ler um registro novamente. Você pode evitar esses problemas definindo a frequência de leitura do consumidor.

Para definir a frequência de leitura do consumidor do Kinesis, você define a configuração `SHARD_GETRECORDS_INTERVAL_MILLIS`.

O exemplo de código a seguir define a configuração `SHARD_GETRECORDS_INTERVAL_MILLIS` para um segundo:

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS,
 "1000");
```

Configure o buffer de gravação

Nesta seção, você define a frequência de gravação e outras configurações do coletor.

Por padrão, o aplicativo grava no bucket de destino a cada minuto. Você pode alterar esse intervalo e outras configurações configurando o objeto `DefaultRollingPolicy`.

> **Note**
>
> O coletor de arquivos de streaming do Apache Flink grava em seu bucket de saída toda vez que o aplicativo cria um ponto de verificação. Por padrão, o aplicativo cria um ponto de

verificação a cada minuto. Para aumentar o intervalo de gravação do coletor do S3, você também deve aumentar o intervalo do ponto de verificação.

Para configurar o objeto `DefaultRollingPolicy`, faça o seguinte:

1. Aumente a `CheckpointInterval` configuração do aplicativo. A entrada a seguir para a UpdateApplicationação define o intervalo do ponto de verificação em 10 minutos:

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
         }
      }
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5
}
```

   Para usar o código anterior, especifique a versão atual do aplicativo. Você pode recuperar a versão do aplicativo usando a ListApplicationsação.

2. Adicione a seguinte declaração de importação ao início do arquivo `S3StreamingSinkJob.java`:

```
import java.util.concurrent.TimeUnit;
```

3. Atualize o método `createS3SinkFromStaticConfig` no arquivo `S3StreamingSinkJob.java` para que fique como se segue:

```
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;
    }
```

O exemplo de código anterior define a frequência de gravações no bucket do Amazon S3 em oito minutos.

Para obter mais informações sobre como configurar o coletor de arquivos de streaming do Apache Flink, consulte Formatos codificados por linha na documentação do Apache Flink.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar os recursos AWS que você criou no tutorial do Amazon S3.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seu fluxo de dados do Kinesis
- Exclua seus objetos e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seu fluxo de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream

3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.

Exclua seus objetos e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Políticas.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Funções.
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Na barra de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Tutorial: Usando um aplicativo Managed Service for Apache Flink para replicar dados de um tópico em um cluster MSK para outro em uma VPC

O tutorial a seguir demonstra como criar uma VPC da Amazon com um cluster do Amazon MSK e dois tópicos e como criar um aplicativo Managed Service for Apache Flink que lê um tópico do Amazon MSK e grava em outro.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Introdução (DataStream API).

Este tutorial contém as seguintes seções:

- Crie uma VPC com um cluster do Amazon MSK
- Crie o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar o aplicativo
- Configurar o aplicativo
- Executar o aplicativo
- Testar o aplicativo

Crie uma VPC com um cluster do Amazon MSK

Para criar um exemplo de VPC e de cluster do Amazon MSK para acessar a partir de um aplicativo Managed Service for Apache Flink, siga o tutorial Conceitos básicos do uso do Amazon MSK.

Ao concluir o tutorial, observe o seguinte:

- Na Etapa 3: Crie um tópico, repita o comando `kafka-topics.sh --create` para criar um tópico de destino chamado `AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-
factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

- Registre a lista de servidores bootstrap do seu cluster. Você pode obter a lista de servidores bootstrap com o seguinte comando (*ClusterArn*substitua pelo ARN do seu cluster 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"
```

```
}
```

- Ao seguir as etapas dos tutoriais, certifique-se de usar a região AWS selecionada no código, nos comandos e nas entradas do console.

Crie o código do aplicativo

Nesta seção, você baixará e compilará o arquivo JAR do aplicativo. Recomendamos usar o Java 11.

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. O código do aplicativo está localizado no arquivo `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java`. Você pode examinar o código para se familiarizar com a estrutura do código do aplicativo Managed Service for Apache Flink.
4. Use a ferramenta Maven de linha de comando ou seu ambiente de desenvolvimento preferido para criar o arquivo JAR. Para compilar o arquivo JAR usando a ferramenta Maven de linha de comando, digite o seguinte:

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

Se a compilação for feita com sucesso, o seguinte arquivo será criado:

```
target/KafkaGettingStartedJob-1.0.jar
```

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11. Se você estiver usando um ambiente de desenvolvimento,

faça o upload do código Java Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Introdução (DataStream API).

> **Note**
>
> Se você excluiu o bucket do Amazon S3 no tutorial de introdução, siga a etapa the section called “faça o upload do código Java Apache Flink Streaming” novamente.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `KafkaGettingStartedJob-1.0.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink 1.15.2.
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **`ka-app-code-<username>`**.
   - Em Caminho do objeto do Amazon S3, insira **`KafkaGettingStartedJob-1.0.jar`**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.

   > **Note**
   >
   > Quando você especifica recursos do aplicativo usando o console (como CloudWatch Logs ou uma Amazon VPC), o console modifica sua função de execução do aplicativo para conceder permissão para acessar esses recursos.

4. Em Propriedades, selecione Adicionar grupo. Insira as seguintes propriedades:

ID do grupo	Chave	Valor
`KafkaSource`	tópico	AWSKafkaTutorialTopic
`KafkaSource`	bootstrap.servers	*A lista de servidores bootstrap que você salvou anteriormente*

ID do grupo	Chave	Valor
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

> **Note**
>
> O ssl.truststore.password para o certificado padrão é “changeit”; você não precisa alterar esse valor se estiver usando o certificado padrão.

Selecione Adicionar grupo novamente. Insira as seguintes propriedades:

ID do grupo	Chave	Valor
KafkaSink	tópico	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	*A lista de servidores bootstrap que você salvou anteriormente*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

O código do aplicativo lê as propriedades do aplicativo acima para configurar a origem e o coletor usados para interagir com sua VPC e com o cluster do Amazon MSK. Para obter mais informações sobre usar as propriedades, consulte Propriedades de runtime.

5. Em Snapshots, selecione Desativar. Isso facilitará a atualização do aplicativo sem carregar dados inválidos do estado do aplicativo.
6. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
7. Para CloudWatch registrar, escolha a caixa de seleção Ativar.
8. Na seção Nuvem privada virtual (VPC), selecione a VPC a ser associada ao aplicativo. Selecione as sub-redes e o grupo de segurança associados à sua VPC os quais você deseja que o aplicativo use para acessar os recursos da VPC.
9. Selecione Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`
>
> Esse fluxo de logs é usado para monitorar o aplicativo.

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Testar o aplicativo

Nesta seção, você grava registros no tópico de origem. O aplicativo lê registros do tópico de origem e os grava no tópico de destino. Você verifica se o aplicativo está funcionando gravando registros no tópico de origem e lendo registros do tópico de destino.

Para escrever e ler registros dos tópicos, siga as etapas de Etapa 6: Produza e consuma dados no tutorial de Introdução ao uso do Amazon MSK.

Para ler o tópico de destino, use o nome do tópico de destino em vez do nome do tópico de origem em sua segunda conexão com o cluster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Se nenhum registro aparecer no tópico de destino, consulte a seção Não é possível acessar recursos em uma VPC no tópico Solução de problemas.

Exemplo: Use um consumidor EFO com um fluxo de dados do Kinesis

Neste exercício, você cria um aplicativo Managed Service for Apache Flink que lê a partir de um fluxo de dados do Kinesis usando um consumidor Enhanced Fan-Out (EFO). Se um consumidor do Kinesis usa o EFO, o serviço Kinesis Data Streams fornece sua própria largura de banda dedicada, em vez de fazer com que o consumidor compartilhe a largura de banda fixa do stream com os outros consumidores que estão lendo o stream.

Para obter mais informações sobre como usar o EFO com o consumidor Kinesis, consulte FLIP-128: distribuição avançada para consumidores da Kinesis.

O aplicativo que você cria neste exemplo usa o AWS Kinesis Connector (flink-connector-kinesis) 1.15.3.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Introdução (DataStream API).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Compilar o código do aplicativo

- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seu fluxo de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/EfoConsumer`.

O código do aplicativo está localizado no arquivo `EfoApplication.java`. Observe o seguinte sobre o código do aplicativo:

- Você habilita o consumidor EFO definindo os seguintes parâmetros no consumidor do Kinesis:
  - RECORD_PUBLISHER_TYPE: defina esse parâmetro como EFO para que seu aplicativo use um consumidor EFO para acessar os dados do Kinesis Data Stream.
  - EFO_CONSUMER_NAME: defina esse parâmetro como um valor de sequência de caracteres que é exclusivo entre os consumidores desse fluxo. A reutilização de um nome de consumidor no mesmo Kinesis Data Stream fará com que o consumidor anterior que usava esse nome seja excluído.
- O exemplo de código a seguir demonstra como atribuir valores às propriedades de configuração do consumidor para usar um consumidor EFO para ler o fluxo de origem:

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

Compilar o código do aplicativo

Para compilar o aplicativo, faça o seguinte:

1. Instale o Java e o Maven, caso ainda não o tenha feito. Para obter mais informações, consulte Pré-requisitos no tutorial Introdução (DataStream API).
2. Compile o aplicativo com o seguinte comando:

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

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

A compilação do aplicativo cria o arquivo JAR do aplicativo (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

faça o upload do código Java Apache Flink Streaming

Nesta seção, você o faz upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

> **Note**
>
> Essas permissões concedem ao aplicativo a capacidade de acessar o consumidor 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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.

4. Em Propriedades, selecione Criar grupo.
5. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
ConsumerConfigProperties	**flink.stream.recordpublisher**	**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. Em Propriedades, selecione Criar grupo.
7. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
ProducerConfigProperties	**OUTPUT_STREAM**	**ExampleOutputStream**
ProducerConfigProperties	**AWS_REGION**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

8. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
9. Para CloudWatch registrar, marque a caixa de seleção Ativar.
10. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`
>
> Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console para verificar se o aplicativo está funcionando.

Você também pode conferir o nome do seu consumidor () no console do Kinesis Data Streams, na guia Enhanced fan-out do stream de dados. basic-efo-flink-app

Limpar AWS recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Janela EFO.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink

2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.

4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: Gravando no Kinesis Data Firehose

Neste exercício, você cria um aplicativo Managed Service for Apache Flink que tem um fluxo de dados do Kinesis como fonte e um fluxo do Kinesis Data Firehose como coletor. Usando o coletor, você pode conferir a saída do aplicativo em um bucket do Amazon S3.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Introdução (DataStream API).

Esta seção contém as seguintes etapas:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixar e examinar o código Java Apache Flink Streaming
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Um fluxo de dados do Kinesis (`ExampleInputStream`)
- Um fluxo do Kinesis Data Firehose no qual o aplicativo grava o resultado (`ExampleDeliveryStream`).
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

É possível criar o fluxo de dados do Kinesis, os buckets do Amazon S3 e o fluxo do Kinesis Data Firehose usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seu fluxo de dados **`ExampleInputStream`**.
- Criando um fluxo de entrega do Amazon Kinesis Firehose Data no Guia do desenvolvedor do Amazon Kinesis Data Firehose. Nomeie o fluxo do Kinesis Data Firehose **`ExampleDeliveryStream`**. Ao criar o fluxo do Kinesis Data Firehose, crie também o destino no S3 e o perfil do IAM do fluxo.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/FirehoseSink`.

O código do aplicativo está localizado no arquivo `FirehoseSinkStreamingJob.java`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
```

```
                new SimpleStringSchema(), inputProperties));
```

- O aplicativo usa um coletor do Kinesis Data Firehose para gravar dados em um fluxo do Kinesis Data Firehose. O trecho a seguir cria o coletor do Kinesis Data Firehose sink:

```
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 o código do aplicativo

Para compilar o aplicativo, faça o seguinte:

1. Instale o Java e o Maven, caso ainda não o tenha feito. Para obter mais informações, consulte Pré-requisitos no tutorial Introdução (DataStream API).
2. Para usar o conector Kinesis no aplicativo a seguir, você precisa baixar, compilar e instalar o Apache Maven. Para obter mais informações, consulte the section called “Usar o conector de fluxos Kinesis para o Apache Flink com versões anteriores do Apache Flink”.
3. Compile o aplicativo com o seguinte comando:

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

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11.

A compilação do aplicativo cria o arquivo JAR do aplicativo (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

faça o upload do código Java Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. No console, escolha o ka-app-code- <username>bucket e, em seguida, escolha Upload.
3. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `java-getting-started-1.0.jar` que você criou na etapa anterior.
4. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI.

> **Note**
>
> Quando você cria o aplicativo usando o console, seus recursos AWS Identity and Access Management (IAM) e do Amazon CloudWatch Logs são criados para você. Quando cria o aplicativo usando a AWS CLI, você cria esses recursos separadamente.

Tópicos

- Criar e executar o aplicativo (console)
- Criar e executar o aplicativo (AWS CLI)

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Quando você cria o aplicativo usando o console, você tem a opção de criar um perfil e uma política do IAM para o seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso ao fluxo de dados do Kinesis e ao fluxo do Kinesis Data Firehose.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.

2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua todas as ocorrências do exemplo de IDs de conta (*012345678901*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **java-getting-started-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
5. Para CloudWatch registrar, marque a caixa de seleção Ativar.

6. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualize o aplicativo

Usando o console, você pode atualizar configurações do aplicativo, como as propriedades do aplicativo, as configurações de monitoramento e a localização ou o nome do arquivo JAR do aplicativo.

Na MyApplicationpágina, escolha Configurar. Atualize as configurações do aplicativo e selecione Update (Atualizar).

> **Note**
>
> Para atualizar o código do aplicativo no console, você deve alterar o nome do objeto do JAR, usar um bucket do S3 diferente ou usar o AWS CLI conforme descrito na seção the section called “Atualizar o código do aplicativo”. Se o nome do arquivo ou o bucket não mudar, o código do aplicativo não será recarregado quando você selecionar Atualizar na página Configure.

Criar e executar o aplicativo (AWS CLI)

Nesta seção, você usa a AWS CLI para criar e executar o aplicativo Flink do Managed Service for Apache Flink.

Criar uma política de permissões

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação `read` no fluxo de origem, e outra que concede permissões para ações `write` no fluxo de destino. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua *`username`* pelo nome de usuário que você usará para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos Nomes de recurso da Amazon (ARNs) (*`012345678901`*) pelo ID da conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

> **Note**
>
> Para acessar outros serviços da Amazon, você pode usar o AWS SDK for Java. O Managed Service for Apache Flink define automaticamente as credenciais exigidas pelo SDK como as credenciais do perfil do IAM associado a seu aplicativo. Não é necessária nenhuma etapa adicional.

Criar uma perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo se não tiver permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink a permissão de assumir a função. A política de permissões determina o que o Managed Service for Apache Flink pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles (Funções) e Create Role (Criar função).
3. Em Selecionar tipo de identidade de confiança, selecione Serviço da AWS. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis. Em Select your use case (Selecionar seu caso de uso), selecione Kinesis Analytics.

   Selecione Next: Permissions (Próximo: permissões).

4. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
5. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
6. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, the section called “Criar uma política de permissões”.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).
   b. Selecione Attach Policies.
   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).
   d. Escolha a ReadSourceStreamWriteSinkStream política AK e escolha Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usará para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo do Managed Service for Apache Flink

1. Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket pelo sufixo que você selecionou na seçãothe section called “Criar recursos dependentes” (`ka-app-code-<username>`). Substitua o ID da conta de exemplo (*`012345678901`*) na função de execução do serviço pelo ID da conta.

```
{
    "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. Execute a ação `CreateApplication` com a solicitação anterior para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação `StartApplication` para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

```
    }
}
```

2. Execute a ação StartApplication com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação StopApplication para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Execute a ação StopApplication com a seguinte solicitação para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consultethe section called “Configurando o log”.

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação UpdateApplicationAWS CLI.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (<*username*>) com o sufixo que você selecionou na seção the section called “Criar recursos dependentes”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Limpar AWS recursos

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial Introdução.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seu fluxo de dados do Kinesis
- Exclua o fluxo do Kinesis Data Firehose
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Selecione Configurar.
4. Na seção Snapshots, selecione Desativar e, em seguida, selecione Atualizar.
5. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seu fluxo de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.

Exclua o fluxo do Kinesis Data Firehose

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Firehose, escolha. ExampleDeliveryStream
3. Na ExampleDeliveryStreampágina, escolha Excluir stream do Kinesis Data Firehose e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.
4. Se você criou um bucket do Amazon S3 para o destino do fluxo do Kinesis Data Firehose, exclua esse bucket também.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.

4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Se você criou uma nova política para o fluxo do Kinesis Data Firehose, exclua essa política também.
7. Na barra de navegação, selecione Roles (Funções).
8. Escolha a função kinesis-analytics- MyApplication -us-west-2.
9. Selecione Excluir função e, em seguida, confirme a exclusão.
10. Se você criou uma nova função para o fluxo do Kinesis Data Firehose, exclua essa função também.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: Leia a partir de um fluxo de dados do Kinesis em outra conta.

Este exemplo demonstra como criar um Managed Service para o aplicativo Apache Flink que lê dados de um fluxo do Kinesis em uma conta diferente. Neste exemplo, você usará uma conta para o fluxo de origem do Kinesis e uma segunda conta para o aplicativo Managed Service for Apache Flink e para o fluxo de dados do coletor do Kinesis.

Este tópico contém as seguintes seções:

- Pré-requisitos
- Configuração
- Criar fluxo da fonte do Kinesis
- Crie e atualize os perfis e políticas do IAM
- Atualize o script Python
- Atualize o aplicativo Java.
- Compile, faça o upload e execute o aplicativo.

Pré-requisitos

- Neste tutorial, você modifica o exemplo da Introdução para ler dados de um fluxo do Kinesis em uma conta diferente. Conclua o tutorial Introdução (DataStream API) antes de continuar.
- Você precisa de duas AWS contas para concluir este tutorial: uma para o fluxo de origem e outra para o aplicativo e o fluxo do coletor. Use a conta AWS que você usou para o tutorial da Introdução para o aplicativo e o fluxo do coletor. Use uma conta AWS diferente para o fluxo de origem.

Configuração

Você acessará suas duas contas AWS usando perfis nomeados. Modifique suas AWS credenciais e arquivos de configuração para incluir dois perfis que contenham a região e as informações de conexão de suas duas contas.

O arquivo de credencial de exemplo a seguir contém dois perfis nomeados, `ka-source-stream-account-profile` e `ka-sink-stream-account-profile`. Use a conta que você usou no tutorial da Introdução para a conta do fluxo do coletor.

```
[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
```

O arquivo de configuração de exemplo a seguir contém os mesmos perfis nomeados com informações de região e formato de saída.

```
[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**
>
> Este tutorial não usa o `ka-sink-stream-account-profile`. Ele está incluído como um exemplo de como acessar duas contas AWS diferentes usando perfis.

Para obter mais informações sobre perfis nomeados com o AWS CLI, consulte Perfis nomeados na documentação AWS Command Line Interface.

Criar fluxo da fonte do Kinesis

Nesta seção, você criará o fluxo do Kinesis na conta de origem.

Use o comando a seguir para criar o fluxo do Kinesis que o aplicativo usará como entrada. Observe que o parâmetro `--profile` especifica qual perfil de conta usar.

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

Crie e atualize os perfis e políticas do IAM

Para permitir o acesso a objetos em todas as contas AWS, você deve crias um perfil e uma política do IAM na conta de origem. Em seguida, você modifica a política do IAM na conta do coletor. Para obter mais informações sobre como criar perfis e políticas do IAM, consulte os seguintes tópicos no AWS Identity and Access ManagementGuia do usuário:

- Criando perfis do IAM
- Criando políticas do IAM

Perfis e políticas da conta do coletor

1. Edite a `kinesis-analytics-service-MyApplication-us-west-2` política do tutorial da Introdução. Essa política permite que o perfil da conta de origem seja assumido para ler o fluxo de origem.

> **Note**
>
> Quando você usa o console para criar seu aplicativo, o console cria uma política chamada `kinesis-analytics-service-<application name>-<application region>` e um perfil chamado `kinesisanalytics-<application name>-<application region>`.

Adicione a seção destacada abaixo à política. Substitua o exemplo de ID de conta (*SOURCE01234567*) pelo ID da conta que você usará para o fluxo de origem.

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

2. Abra o perfil `kinesis-analytics-MyApplication-us-west-2` e anote o nome do recurso da Amazon (ARN). Ele será necessário na próxima seção. O ARN do perfil é semelhante ao seguinte.

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-
west-2
```

Perfis e políticas da conta de origem

1. Crie uma política na conta de origem chamada `KA-Source-Stream-Policy`. Use o seguinte JSON para a política. Substitua o número da conta de exemplo pelo número da conta de origem.

```
{
    "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. Crie um perfil na conta de origem chamado `MF-Source-Stream-Role`. Faça o seguinte para criar o perfil usando o caso de uso do Managed Flink:

    1. No console de gerenciamento do IAM, selecione Criar perfil.
    2. Na página Criar perfil, selecione AWSServiço. Na lista de serviços, selecione Kinesis.
    3. Na seção Selecione seu caso de uso, selecione Managed Service for Apache Flink.
    4. Selecione Next: Permissions (Próximo: permissões).
    5. Adicione a política de permissões `KA-Source-Stream-Policy` que criada na etapa anterior. Selecione Next: Tags (Próximo: tags).
    6. Selecione Next: Review (Próximo: revisar).
    7. Nomeie a função `KA-Source-Stream-Role`. Seu aplicativo usará esse perfil para acessar o fluxo de origem.

3. Adicione o `kinesis-analytics-MyApplication-us-west-2` ARN da conta do coletor à relação de confiança do `KA-Source-Stream-Role` perfil na conta de origem:

    1. Abra o `KA-Source-Stream-Role` no console do IAM.
    2. Selecione a guia Relacionamentos de confiança.
    3. Selecione Edit trust relationship (Editar relação de confiança).
    4. Use o código a seguir para a relação de confiança. Substitua o exemplo de IDs de conta (*SINK012345678*) pelo ID da conta do coletor.

```
{
```

```
    "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"
      }
    ]
}
```

Atualize o script Python

Nesta seção, você atualiza o script Python que gera dados de amostra para usar no perfil da conta de origem.

Atualize o script `stock.py` com as seguintes alterações destacadas.

```
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")
```

Atualize o aplicativo Java.

Nesta seção, você atualiza o código do aplicativo Java para assumir a função da conta de origem ao ler o fluxo de origem.

Faça as alterações a seguir no arquivo `BasicStreamingJob.java`. Substitua o exemplo do número da conta de origem (*SOURCE01234567*) pelo número da conta de origem.

```
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, faça o upload e execute o aplicativo.

Faça o seguinte para atualizar e executar o aplicativo:

1. Compile o aplicativo novamente executando o comando a seguir no diretório com o arquivo `pom.xml`.

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

2. Exclua o arquivo JAR anterior do seu bucket do Amazon Simple Storage Service (Amazon S3) e, em seguida, faça o upload do novo arquivo `aws-kinesis-analytics-java-apps-1.0.jar` no bucket do Amazon S3.
3. Na página do aplicativo no console Managed Service for Apache Flink, selecione Configurar, Atualizar para recarregar o arquivo JAR do aplicativo.
4. Execute o script `stock.py` para enviar dados para o fluxo de origem.

```
python stock.py
```

Agora, o aplicativo lê dados do fluxo do Kinesis na outra conta.

Você pode ver se o aplicativo está funcionando verificando a métrica `PutRecords.Bytes` do fluxo `ExampleOutputStream`. Se houver atividade no fluxo de saída, o aplicativo está funcionando corretamente.

Tutorial: Usando um Truststore personalizado com o Amazon MSK

APIs de fonte de dados atuais

Se você estiver usando as APIs de fonte de dados atuais, seu aplicativo poderá aproveitar o utilitário MSK Config Providers descrito aqui. Isso permite que sua KafkaSource função acesse seu armazenamento de chaves e armazenamento confiável para TLS mútuo no 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 + "}");
...
```

Mais detalhes e uma explicação passo a passo podem ser encontrados aqui.

SourceFunction APIs legadas

Se você estiver usando as SourceFunction APIs legadas, seu aplicativo usará esquemas personalizados de serialização e desserialização que substituem o `open` método para carregar o armazenamento confiável personalizado. Isso torna o truststore disponível para o aplicativo após o aplicativo ser reiniciado ou substituído pelos encadeamentos.

O truststore personalizado é recuperado e armazenado usando o seguinte 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 Kakfa truststore", ex);
    }
}
```

> **Note**
>
> O Apache Flink exige que o truststore esteja no formato JKS.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, em primeiro lugar conclua o exercício. Introdução (DataStream API)

O tutorial a seguir demonstra como se conectar com segurança (criptografia em trânsito) a um cluster do Kafka que usa certificados de servidor emitidos por uma Autoridade Certificadora (AC) personalizada, privada ou, até mesmo, auto-hospedada.

Para conectar qualquer cliente do Kafka de forma segura via TLS a um cluster do Kafka, o cliente do Kafka (como o exemplo do aplicativo Flink) deve confiar em toda a cadeia de confiança apresentada pelos certificados de servidor do cluster do Kafka (da AC emissora até a AC de nível raiz). Como exemplo de um truststore personalizado, usaremos um cluster do Amazon MSK com a autenticação TLS mútua (MTLS) habilitada. Isso significa que os nós do cluster do MSK usam certificados de servidor emitidos por uma Autoridade de Certificação Privada do AWSCertificate Manager que é privada na sua conta e região e, portanto, não é confiável para o Truststore padrão do Java Virtual Machine (JVM) que executa o aplicativo Flink.

> **Note**
>
> - Um Keystore é usado para armazenar a chave privada e os certificados de identidade que um aplicativo deve apresentar ao servidor ou ao cliente para verificação.

- Um Truststore é usado para armazenar certificados de Autoridades Certificadas (AC) que verificam o certificado apresentado pelo servidor em uma conexão SSL.

Você também pode usar a técnica deste tutorial para interações entre um aplicativo Managed Service for Apache Flink e outras fontes do Apache Kafka, como:

- Um cluster personalizado do Apache Kafka hospedado em AWS (Amazon EC2 ou Amazon EKS)
- Um cluster do Confluent Kafka hospedado em AWS
- Um cluster on-premises do Kafka acessado por meio de AWS Direct Connect ou uma VPN

Este tutorial contém as seguintes seções:

- Crie uma VPC com um cluster do Amazon MSK
- Crie um truststore personalizado e aplique-o ao seu cluster
- Crie o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar o aplicativo
- Configurar o aplicativo
- Executar o aplicativo
- Testar o aplicativo

Crie uma VPC com um cluster do Amazon MSK

Para criar um exemplo de VPC e de cluster do Amazon MSK para acessar a partir de um aplicativo Managed Service for Apache Flink, siga o tutorial Conceitos básicos do uso do Amazon MSK.

Ao concluir o tutorial, faça também o seguinte:

- Na Etapa 3: Crie um tópico, repita o comando `kafka-topics.sh --create` para criar um tópico de destino chamado `AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --
replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

> **Note**
>
> Se o comando `kafka-topics.sh` retornar um `ZooKeeperClientTimeoutException`, verifique se o grupo de segurança do cluster do Kafka tem uma regra de entrada para permitir todo o tráfego do endereço IP privado da instância do cliente.

- Registre a lista de servidores bootstrap do seu cluster. Você pode obter a lista de servidores bootstrap com o seguinte comando (*`ClusterArn`*substitua pelo ARN do seu cluster 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"
}
```

- Ao seguir as etapas deste tutorial e dos tutoriais de pré-requisito, certifique-se de usar a região AWS selecionada no seu código, nos comandos e nas entradas do console.

Crie um truststore personalizado e aplique-o ao seu cluster

Nesta seção, você cria uma autoridade de certificação (AC) personalizada, a usa para gerar um truststore personalizado e a aplica ao seu cluster do MSK.

Para criar e aplicar seu truststore personalizado, siga o tutorial de Autenticação do cliente no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.

Crie o código do aplicativo

Nesta seção, você baixará e compilará o arquivo JAR do aplicativo.

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. O código do aplicativo está localizado no `amazon-kinesis-data-analytics-java-examples/CustomKeystore`. Você pode examinar o código para se familiarizar com a estrutura do código do Managed Service for Apache Flink.
4. Use a ferramenta Maven de linha de comando ou seu ambiente de desenvolvimento preferido para criar o arquivo JAR. Para compilar o arquivo JAR usando a ferramenta Maven de linha de comando, insira o seguinte:

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

   Se a compilação for feita com sucesso, o seguinte arquivo será criado:

   ```
   target/flink-app-1.0-SNAPSHOT.jar
   ```

   > **Note**
   >
   > O código-fonte fornecido depende de bibliotecas do Java 11.

faça o upload do código Java Apache Flink Streaming

Nesta seção, você faz o upload do código do seu aplicativo no bucket do Amazon S3 que você criou no tutorial Introdução (DataStream API).

> **Note**
>
> Se você excluiu o bucket do Amazon S3 no tutorial de introdução, siga a etapa the section called “faça o upload do código Java Apache Flink Streaming” novamente.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `flink-app-1.0-SNAPSHOT.jar` que você criou na etapa anterior.
3. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink 1.15.2.
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **`ka-app-code-`*`<username>`***.
   - Em Caminho do objeto do Amazon S3, insira **`flink-app-1.0-SNAPSHOT.jar`**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.

> **Note**
>
> Quando você especifica recursos do aplicativo usando o console (como Logs ou uma VPC), o console modifica sua função de execução do aplicativo para conceder permissão para acessar esses recursos.

4. Em Propriedades, selecione Adicionar grupo. Insira as seguintes propriedades:

ID do grupo	Chave	Valor
KafkaSource	tópico	AWSKafkaTutorialTopic
KafkaSource	bootstrap.servers	*A lista de servidores bootstrap que você salvou anteriormente*
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

> **Note**
>
> O ssl.truststore.password para o certificado padrão é “changeit”; você não precisa alterar esse valor se estiver usando o certificado padrão.

Selecione Adicionar grupo novamente. Insira as seguintes propriedades:

ID do grupo	Chave	Valor
KafkaSink	tópico	AWSKafkaTutorialTopicDestination

ID do grupo	Chave	Valor
KafkaSink	bootstrap.servers	*A lista de servidores bootstrap que você salvou anteriormente*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

O código do aplicativo lê as propriedades do aplicativo acima para configurar a origem e o coletor usados para interagir com sua VPC e com o cluster do Amazon MSK. Para obter mais informações sobre usar as propriedades, consulte Propriedades de runtime.

5. Em Snapshots, selecione Desativar. Isso facilitará a atualização do aplicativo sem carregar dados inválidos do estado do aplicativo.
6. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
7. Para CloudWatch registrar, escolha a caixa de seleção Ativar.
8. Na seção Nuvem privada virtual (VPC), selecione a VPC a ser associada ao aplicativo. Selecione as sub-redes e o grupo de segurança associados à sua VPC os quais você deseja que o aplicativo use para acessar os recursos da VPC.
9. Selecione Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Esse fluxo de logs é usado para monitorar o aplicativo.

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Testar o aplicativo

Nesta seção, você grava registros no tópico de origem. O aplicativo lê registros do tópico de origem e os grava no tópico de destino. Você verifica se o aplicativo está funcionando gravando registros no tópico de origem e lendo registros do tópico de destino.

Para escrever e ler registros dos tópicos, siga as etapas de Etapa 6: Produza e consuma dados no tutorial de Introdução ao uso do Amazon MSK.

Para ler o tópico de destino, use o nome do tópico de destino em vez do nome do tópico de origem em sua segunda conexão com o cluster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Se nenhum registro aparecer no tópico de destino, consulte a seção Não é possível acessar recursos em uma VPC no tópico Solução de problemas.

Exemplos de Python

Os exemplos a seguir demonstram como criar aplicativos usando o Python com a API de tabelas do Apache Flink.

Tópicos

- Exemplo: Criando uma janela em cascata em Python
- Exemplo: Criando uma janela em cascata em Python
- Exemplo: envie dados de streaming para o Amazon S3 em Python

Exemplo: Criando uma janela em cascata em Python

Neste exercício, você cria um aplicativo Python Managed Service for Apache Flink que agrega dados usando uma janela em cascata.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Conceitos básicos (Python).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Comprima e faça upload do código Python do Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seus fluxos de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

> **Note**
>
> O script do Python nesta seção usa o AWS CLI. Você deve configurar seu AWS CLI para usar as credenciais da sua conta e a região padrão. Para configurar o seu AWS CLI, digite o seguinte:
>
> ```
> aws configure
> ```

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

   Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/python/TumblingWindow`.

O código do aplicativo está localizado no arquivo `tumbling-windows.py`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem de tabela do Kinesis para ler o fluxo de origem. O trecho a seguir chama a função `create_table` para criar a origem de tabela Kinesis:

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

A função `create_table` usa um comando SQL para criar uma tabela que é apoiada pela origem de transmissão:

```
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)
```

- O aplicativo usa o operador `Tumble` para agregar registros em uma janela em cascata especificada e retornar os registros agregados como um objeto de tabela:

```
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")
```

- O aplicativo usa o conector Kinesis Flink, do `flink-sql-connector-kinesis-1.15.2.jar`.

Comprima e faça upload do código Python do Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. Use seu aplicativo de compressão preferido para comprimir os arquivos `tumbling-windows.py` e `flink-sql-connector-kinesis-1.15.2.jar`. Nomeie o arquivo como `myapp.zip`.

2. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
3. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `myapp.zip` que você criou na etapa anterior.
4. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa

essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:

- Política: `kinesis-analytics-service-MyApplication-us-west-2`
- Função: `kinesisanalytics-MyApplication-us-west-2`

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **`ka-app-code-<username>`**.
   - Em Caminho do objeto do Amazon S3, insira **`myapp.zip`**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
`consumer.config.0`	**`input.stream.name`**	**`ExampleInputStream`**
`consumer.config.0`	**`aws.region`**	**`us-west-2`**
`consumer.config.0`	**`scan.stream.initpos`**	**`LATEST`**

   Selecione Save (Salvar).
6. Em Propriedades, selecione Adicionar grupo novamente.
7. Insira o seguinte:

ID do grupo	Chave	Valor
`producer.config.0`	**`output.stream.name`**	**`ExampleOutputStream`**
`producer.config.0`	**`aws.region`**	**`us-west-2`**

ID do grupo	Chave	Valor
producer.config.0	**shard.count**	**1**

8. Em Propriedades, selecione Adicionar grupo novamente. Em ID do grupo, digite **kinesis.analytics.flink.run.options**. Esse grupo de propriedades especiais informa ao aplicativo onde encontrar seus recursos de código. Para ter mais informações, consulte Especificar seus arquivos de código.
9. Insira o seguinte:

ID do grupo	Chave	Valor
kinesis.analytics.flink.run.options	**python**	**tumbling-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.jar**

10. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
11. Para CloudWatch registrar, marque a caixa de seleção Ativar.
12. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`
>
> Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console para verificar se o aplicativo está funcionando.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janela em cascata.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis

- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.

8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: Criando uma janela em cascata em Python

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Conceitos básicos (Python).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Comprima e faça upload do código Python do Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar os fluxos do Kinesis e o bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seus fluxos de dados **`ExampleInputStream`** e **`ExampleOutputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

> **Note**
>
> O script do Python nesta seção usa o AWS CLI. Você deve configurar seu AWS CLI para usar as credenciais da sua conta e a região padrão. Para configurar o seu AWS CLI, digite o seguinte:
>
> ```
> aws configure
> ```

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

   Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/>amazon-kinesis-data-analytics-java-
examples
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/python/SlidingWindow`.

O código do aplicativo está localizado no arquivo `sliding-windows.py`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem de tabela do Kinesis para ler o fluxo de origem. O trecho a seguir chama a função `create_input_table` para criar a origem de tabela Kinesis:

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  A função `create_input_table` usa um comando SQL para criar uma tabela que é apoiada pela origem de transmissão:

```
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)
 }
```

- O aplicativo usa o operador `Slide` para agregar registros em uma janela deslizante especificada e retornar os registros agregados como um objeto de tabela:

```
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")
    )
```

- O aplicativo usa o conector Kinesis Flink, do arquivo -1.15.2.jar. flink-sql-connector-kinesis

Comprima e faça upload do código Python do Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

Esta seção descreve como empacotar seu aplicativo Python.

1. Use seu aplicativo de compressão preferido para comprimir os arquivos `sliding-windows.py` e `flink-sql-connector-kinesis-1.15.2.jar`. Nomeie o arquivo como `myapp.zip`.
2. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
3. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `myapp.zip` que você criou na etapa anterior.
4. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.

> **Note**
>
> O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

- Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).

4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **myapp.zip**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
consumer.config.0	**input.stream.name**	**ExampleInputStream**

ID do grupo	Chave	Valor
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

Selecione Save (Salvar).

6. Em Propriedades, selecione Adicionar grupo novamente.
7. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Em Propriedades, selecione Adicionar grupo novamente. Em ID do grupo, digite **kinesis.analytics.flink.run.options**. Esse grupo de propriedades especiais informa ao aplicativo onde encontrar seus recursos de código. Para ter mais informações, consulte Especificar seus arquivos de código.
9. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
kinesis.analytics.flink.run.options	**python**	**sliding-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis_1.15.2.jar**

10. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
11. Para CloudWatch registrar, marque a caixa de seleção Ativar.
12. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`
>
> Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console para verificar se o aplicativo está funcionando.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janelas deslizantes.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.

2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: envie dados de streaming para o Amazon S3 em Python

Neste exercício, você cria um aplicativo Managed Service for Apache Flink em Python que transmite dados para um coletor do Amazon Simple Storage Service.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Conceitos básicos (Python).

Este tópico contém as seguintes seções:

- Criar recursos dependentes
- Gravação de registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Comprima e faça upload do código Python do Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Limpar AWS recursos

Criar recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Um fluxo de dados do Kinesis (`ExampleInputStream`)
- Um bucket do Amazon S3 para armazenar o código e saída do aplicativo (`ka-app-code-<username>`)

> **Note**
>
> O Managed Service for Apache Flink não pode gravar dados no Amazon S3 com a criptografia do lado do servidor habilitada no Managed Service for Apache Flink.

Você pode criar o fluxo de dados do Kinesis e um bucket do Amazon S3 usando o console. Para obter instruções sobre como criar esses recursos, consulte os tópicos a seguir:

- Criando e atualizando fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Nomeie seu fluxo de dados **`ExampleInputStream`**.
- Para obter instruções, consulte Como criar um bucket do S3? no Guia do usuário do Amazon Simple Storage Service. Dê ao bucket do Amazon S3 um nome globalmente exclusivo anexando seu nome de login, como **`ka-app-code-<username>`**.

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

> **Note**
>
> O script do Python nesta seção usa o AWS CLI. Você deve configurar seu AWS CLI para usar as credenciais da sua conta e a região padrão. Para configurar o seu AWS CLI, digite o seguinte:
>
> ```
> aws configure
> ```

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o restante do tutorial.

Baixe e examine o código do aplicativo

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/python/S3Sink`.

O código do aplicativo está localizado no arquivo `streaming-file-sink.py`. Observe o seguinte sobre o código do aplicativo:

- O aplicativo usa uma origem de tabela do Kinesis para ler o fluxo de origem. O trecho a seguir chama a função `create_source_table` para criar a origem de tabela Kinesis:

```
table_env.execute_sql(
        create_source_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

A função `create_source_table` usa um comando SQL para criar uma tabela que é apoiada pela fonte 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'))
```

- O aplicativo usa o conector `filesystem` para enviar registros para um bucket do 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)
```

- O aplicativo usa o conector Kinesis Flink, do arquivo -1.15.2.jar. flink-sql-connector-kinesis

Comprima e faça upload do código Python do Apache Flink Streaming

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Criar recursos dependentes.

1. Use seu aplicativo de compactação preferido para compactar os arquivos `streaming-file-sink.py` e flink-sql-connector-kinesis-1.15.2.jar. Nomeie o arquivo como `myapp.zip`.
2. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
3. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `myapp.zip` que você criou na etapa anterior.
4. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.15.2.

   - Deixe o menu suspenso de versão como Apache Flink versão 1.15.2 (versão recomendada).

4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **myapp.zip**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

Selecione Save (Salvar).

6. Em Propriedades, selecione Adicionar grupo novamente. Em ID do grupo, digite **`kinesis.analytics.flink.run.options`**. Esse grupo de propriedades especiais informa ao aplicativo onde encontrar seus recursos de código. Para ter mais informações, consulte Especificar seus arquivos de código.
7. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	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`**

8. Em Propriedades, selecione Adicionar grupo novamente. Em ID do grupo, digite **`sink.config.0`**. Esse grupo de propriedades especiais informa ao aplicativo onde encontrar seus recursos de código. Para ter mais informações, consulte Especificar seus arquivos de código.
9. Insira as seguintes propriedades e valores do aplicativo: (substitua o *`bucket-name`* pelo nome real do seu bucket do Amazon S3).

ID do grupo	Chave	Valor
`sink.config.0`	**`output.bucket.name`**	***`bucket-name`***

10. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
11. Para CloudWatch registrar, marque a caixa de seleção Ativar.
12. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, o Managed Service for Apache Flink cria um grupo de registros e um fluxo de registros para você. Os nomes desses recursos são os seguintes:

- Grupo de logs: `/aws/kinesis-analytics/MyApplication`
- Fluxo de logs: `kinesis-analytics-log-stream`

Esse fluxo de logs é usado para monitorar o aplicativo. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

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

```
    ]
}
```

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console para verificar se o aplicativo está funcionando.

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janelas deslizantes.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seu fluxo de dados do Kinesis
- Exclua seus objetos e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seu fluxo de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.

Exclua seus objetos e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo do Scala

Os exemplos a seguir demonstram como criar aplicativos usando o Python com a API de tabelas do Apache Flink.

Tópicos

Exemplo: Criando uma janela em cascata no Scala

> **Note**
>
> A partir da versão 1.15, o Flink não tem mais o Scala. Agora, os aplicativos podem usar a API Java de qualquer versão do Scala. O Flink ainda usa o Scala em alguns componentes importantes internamente, mas não expõe o Scala no carregador de classes do código do usuário. Por isso, os usuários precisam adicionar dependências do Scala em seus arquivos jar.
>
> Para obter mais informações sobre as mudanças do Scala no Flink 1.15, consulte Sem o Scala na versão 1.15.

Neste exercício, você criará um aplicativo de streaming simples que usa o Scala 3.2.0 e a API Java do Flink. DataStream O aplicativo lê os dados do fluxo do Kinesis, os agrega usando janelas deslizantes e grava os resultados no fluxo de saída do Kinesis.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, conclua primeiro o exercício Introdução (Scala).

Este tópico contém as seguintes seções:

- Baixe e examine o código do aplicativo
- Compile e faça o upload do código do aplicativo
- Criar e executar o aplicativo (console)
- Crie e execute o aplicativo (CLI)
- Atualizar o código do aplicativo
- Limpar AWS recursos

Baixe e examine o código do aplicativo

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo `build.sbt` contém informações sobre a configuração e as dependências do aplicativo, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.scala` contém o método principal que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do 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)
}
```

  O aplicativo também usa um coletor do Kinesis para gravar no fluxo de resultados. O trecho a seguir cria o coletor do 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
}
```

- O aplicativo usa o operador de janela para encontrar a contagem de valores para cada símbolo de ação em uma janela em cascata de cinco segundos. O código a seguir cria o operador e envia os dados agregados para um novo coletor de fluxo de dados do 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)
```

- O aplicativo cria conectores de origem e coletor para acessar recursos externos usando um StreamExecutionEnvironment objeto.
- O aplicativo cria conectores de origem e de coletores usando propriedades dinâmicas do aplicativo. As propriedades de runtime do aplicativo para ler e configurar os conectores. Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compile e faça o upload do código do aplicativo

Nesta seção, você compila e faz o upload do código do aplicativo para um bucket do Amazon S3.

Compilar o código do aplicativo

Use a ferramenta de construção SBT para criar o código Scala para o aplicativo. Para instalar o SBT, consulte Instalar o sbt com a configuração cs. Você também precisa instalar o Java Development Kit (JDK). Consulte Pré-requisitos para concluir os exercícios.

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Você pode compilar e empacotar seu código com o SBT:

```
sbt assembly
```

2. Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

Faça o upload do código Scala do Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon S3 e faz upload do código do seu aplicativo.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira `ka-app-code-<username>` no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configurar opções, mantenha as configurações como estão e selecione Próximo.
5. Na etapa Definir permissões, mantenha as configurações como estão e selecione Próximo.
6. Selecione Criar bucket.
7. Selecione o bucket `ka-app-code-<username>` e, em seguida selecione Upload.
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `tumbling-window-scala-1.0.jar` que você criou na etapa anterior.
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:

- Em Nome do aplicativo, insira **MyApplication**.
- Em Descrição, insira **My Scala test app**.
- Em Runtime, selecione Apache Flink.
- Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).

4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Configurar o aplicativo

Siga o procedimento a seguir para configurar o aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **tumbling-window-scala-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

Selecione Save (Salvar).

6. Em Propriedades, selecione Adicionar grupo novamente.
7. Insira o seguinte:

ID do grupo	Chave	Valor
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
9. Para CloudWatch registrar, escolha a caixa de seleção Ativar.
10. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`

- Fluxo de logs: `kinesis-analytics-log-stream`

Editar a política do IAM

Edite a política do IAM para adicionar permissões para acessar o bucket do Amazon S3.

Editar a política do IAM para adicionar permissões do bucket do S3

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa a AWS Command Line Interface para criar e executar o aplicativo Flink do Managed Service for Apache Flink. Use o comando AWS CLI kinesisanalyticsv2 para criar e interagir com os aplicativos Managed Service para aplicativos Apache Flink.

Criação de uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação de ler no fluxo de origem, e outra que concede permissões para ações de gravação no fluxo de coleta. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua **`username`** pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos nomes do recurso da Amazon Resources Names (ARNs) **`(012345678901)`**pelo ID da sua conta. A função **`MF-stream-rw-role`** de execução do serviço deve ser adaptada à função específica do 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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

Criar um perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas

anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Perfis e Criar perfil.
3. Em Selecionar tipo de identidade de confiança, selecione AWSServiço.
4. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis.
5. Em Selecione seu caso de uso, selecione Managed Service for Apache Flink.
6. Selecione Next: Permissions (Próximo: permissões).
7. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
8. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
9. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, Crie uma política de permissões.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).
   b. Selecione Attach Policies.
   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).

d. Selecione a política `AKReadSourceStreamWriteSinkStream` e selecione Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo

Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (nome do usuário) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (012345678901) na função de execução do serviço pelo ID da conta. O `ServiceExecutionRole` deve incluir o perfil do usuário do IAM que você criou na seção 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"
      }
   ]
}
```

Execute o CreateApplicationcom a seguinte solicitação para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação StartApplication para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação StopApplication para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "tumbling_window"
}
```

2. Execute a ação `StopApplication` com a solicitação anterior para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consulte Como configurar o registro de aplicativos.

Atualize as propriedades do ambiente

Nesta seção, você usa a ação UpdateApplication para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote de código, use a ação da UpdateApplicationCLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (<username>) com o sufixo que você selecionou na seção Criar recursos dependentes.

```
{
    "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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janela em cascata.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis

- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.

8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: Criando uma janela em cascata em Python

> **Note**
>
> A partir da versão 1.15, o Flink não tem mais o Scala. Agora, os aplicativos podem usar a API Java de qualquer versão do Scala. O Flink ainda usa o Scala em alguns componentes importantes internamente, mas não expõe o Scala no carregador de classes do código do usuário. Por isso, os usuários precisam adicionar dependências do Scala em seus arquivos jar.
> Para obter mais informações sobre as mudanças do Scala no Flink 1.15, consulte Sem o Scala na versão 1.15.

Neste exercício, você criará um aplicativo de streaming simples que usa o Scala 3.2.0 e a API Java do Flink. DataStream O aplicativo lê os dados do fluxo do Kinesis, os agrega usando janelas deslizantes e grava os resultados no fluxo de saída do Kinesis.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, conclua primeiro o exercício Introdução (Scala).

Este tópico contém as seguintes seções:

- Baixe e examine o código do aplicativo
- Compile e faça o upload do código do aplicativo

- Criar e executar o aplicativo (console)
- Crie e execute o aplicativo (CLI)
- Atualizar o código do aplicativo
- Limpar AWS recursos

Baixe e examine o código do aplicativo

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo `build.sbt` contém informações sobre a configuração e as dependências do aplicativo, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.scala` contém o método principal que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do 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)
}
```

O aplicativo também usa um coletor do Kinesis para gravar no fluxo de resultados. O trecho a seguir cria o coletor do 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
}
```

- O aplicativo usa o operador de janela para encontrar a contagem de valores para cada símbolo de ação em uma janela de dez segundos que desliza por cinco segundos. O código a seguir cria o operador e envia os dados agregados para um novo coletor de fluxo de dados do 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)
```

- O aplicativo cria conectores de origem e coletor para acessar recursos externos usando um StreamExecutionEnvironment objeto.
- O aplicativo cria conectores de origem e de coletores usando propriedades dinâmicas do aplicativo. As propriedades de runtime do aplicativo para ler e configurar os conectores. Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compile e faça o upload do código do aplicativo

Nesta seção, você compila e faz o upload do código do aplicativo para um bucket do Amazon S3.

Compilar o código do aplicativo

Use a ferramenta de construção SBT para criar o código Scala para o aplicativo. Para instalar o SBT, consulte Instalar o sbt com a configuração cs. Você também precisa instalar o Java Development Kit (JDK). Consulte Pré-requisitos para concluir os exercícios.

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Você pode compilar e empacotar seu código com o SBT:

```
sbt assembly
```

2. Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

Faça o upload do código Scala do Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon S3 e faz upload do código do seu aplicativo.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira `ka-app-code-<username>` no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configurar opções, mantenha as configurações como estão e selecione Próximo.
5. Na etapa Definir permissões, mantenha as configurações como estão e selecione Próximo.
6. Selecione Criar bucket.
7. Selecione o bucket `ka-app-code-<username>` e, em seguida selecione Upload.
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `sliding-window-scala-1.0.jar` que você criou na etapa anterior.
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My Scala test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

Siga o procedimento a seguir para configurar o aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **sliding-window-scala-1.0.jar.**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Propriedades, selecione Adicionar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Selecione Save (Salvar).
6. Em Propriedades, selecione Adicionar grupo novamente.
7. Insira o seguinte:

ID do grupo	Chave	Valor
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
9. Para CloudWatch registrar, escolha a caixa de seleção Ativar.
10. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Editar a política do IAM

Edite a política do IAM para adicionar permissões para acessar o bucket do Amazon S3.

Editar a política do IAM para adicionar permissões do bucket do S3

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*`012345678901`*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa a AWS Command Line Interface para criar e executar o aplicativo Flink do Managed Service for Apache Flink. Use o comando AWS CLI kinesisanalyticsv2 para criar e interagir com os aplicativos Managed Service para aplicativos Apache Flink.

Criação de uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação de ler no fluxo de origem, e outra que concede permissões para ações de gravação no fluxo de coleta. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua **`username`** pelo nome de usuário que você

usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos nomes do recurso da Amazon Resources Names (ARNs) **(012345678901)**pelo ID da sua conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

Criar um perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Perfis e Criar perfil.
3. Em Selecionar tipo de identidade de confiança, selecione AWSServiço.
4. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis.
5. Em Selecione seu caso de uso, selecione Managed Service for Apache Flink.
6. Selecione Next: Permissions (Próximo: permissões).
7. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
8. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
9. Anexe a política de permissões à função.

> **Note**
>
> Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, Crie uma política de permissões.

a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).
b. Selecione Attach Policies.
c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).
d. Selecione a política `AKReadSourceStreamWriteSinkStream` e selecione Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo

Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (nome do usuário) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (012345678901) na função de execução do serviço pelo ID da conta.

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

Execute o CreateApplicationcom a seguinte solicitação para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação StartApplication para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação StopApplication para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "sliding_window"
}
```

2. Execute a ação `StopApplication` com a solicitação anterior para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consulte Como configurar o registro de aplicativos.

Atualize as propriedades do ambiente

Nesta seção, você usa a ação UpdateApplication para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote de código, use a ação da UpdateApplicationCLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (<username>) com o sufixo que você selecionou na seção Criar recursos dependentes.

```
{
    "ApplicationName": "sliding_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
```

```
            }
        }
    }
}
```

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janelas deslizantes.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Exemplo: Enviar dados de streaming para o Amazon S3 no Scala

> **Note**
>
> A partir da versão 1.15, o Flink não tem mais o Scala. Agora, os aplicativos podem usar a API Java de qualquer versão do Scala. O Flink ainda usa o Scala em alguns componentes importantes internamente, mas não expõe o Scala no carregador de classes do código do usuário. Por isso, os usuários precisam adicionar dependências do Scala em seus arquivos jar.
> Para obter mais informações sobre as mudanças do Scala no Flink 1.15, consulte Sem o Scala na versão 1.15.

Neste exercício, você criará um aplicativo de streaming simples que usa o Scala 3.2.0 e a API Java do Flink. DataStream O aplicativo lê os dados do fluxo do Kinesis, os agrega usando janelas deslizantes e grava os resultados no S3.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, conclua primeiro o exercício Introdução (Scala). Você só precisa criar uma pasta adicional ***data/*** no bucket ka-app-code do Amazon S3 -. <username>

Este tópico contém as seguintes seções:

- Baixe e examine o código do aplicativo
- Compile e faça o upload do código do aplicativo
- Criar e executar o aplicativo (console)
- Crie e execute o aplicativo (CLI)
- Atualizar o código do aplicativo
- Limpar AWS recursos

Baixe e examine o código do aplicativo

O código do aplicativo Python para este exemplo está disponível em. GitHub Para fazer download do código do aplicativo, faça o seguinte:

1. Instale o cliente do Git se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Git.
2. Duplique o repositório remoto com o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/scala/S3Sink`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo `build.sbt` contém informações sobre a configuração e as dependências do aplicativo, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.scala` contém o método principal que define a funcionalidade do aplicativo.

- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do 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)
}
```

  O aplicativo também usa StreamingFileSink a para gravar em um bucket do 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()
}
```

- O aplicativo cria conectores de origem e coletor para acessar recursos externos usando um StreamExecutionEnvironment objeto.
- O aplicativo cria conectores de origem e de coletores usando propriedades dinâmicas do aplicativo. As propriedades de runtime do aplicativo para ler e configurar os conectores. Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compile e faça o upload do código do aplicativo

Nesta seção, você compila e faz o upload do código do aplicativo para um bucket do Amazon S3.

Compilar o código do aplicativo

Use a ferramenta de construção SBT para criar o código Scala para o aplicativo. Para instalar o SBT, consulte Instalar o sbt com a configuração cs. Você também precisa instalar o Java Development Kit (JDK). Consulte Pré-requisitos para concluir os exercícios.

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Você pode compilar e empacotar seu código com o SBT:

```
sbt assembly
```

2. Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

Faça o upload do código Scala do Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon S3 e faz upload do código do seu aplicativo.

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira `ka-app-code-<username>` no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configurar opções, mantenha as configurações como estão e selecione Próximo.
5. Na etapa Definir permissões, mantenha as configurações como estão e selecione Próximo.
6. Selecione Criar bucket.
7. Selecione o bucket `ka-app-code-<username>` e, em seguida selecione Upload.
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `s3-sink-scala-1.0.jar` que você criou na etapa anterior.
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink

2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe a versão como Apache Flink versão 1.15.2 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configurar o aplicativo

Siga o procedimento a seguir para configurar o aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **`ka-app-code-`***`<username>`*.
   - Em Caminho do objeto do Amazon S3, insira **`s3-sink-scala-1.0.jar`**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.

4. Em Propriedades, selecione Adicionar grupo.
5. Insira o seguinte:

ID do grupo	Chave	Valor
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

Selecione Save (Salvar).

6. Em Propriedades, selecione Adicionar grupo.
7. Insira o seguinte:

ID do grupo	Chave	Valor
ProducerConfigProperties	**s3.sink.path**	**s3a://ka-app-code-*<user-name>*/data**

8. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
9. Para CloudWatch registrar, escolha a caixa de seleção Ativar.
10. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`

- Fluxo de logs: `kinesis-analytics-log-stream`

Editar a política do IAM

Edite a política do IAM para adicionar permissões para acessar o bucket do Amazon S3.

Editar a política do IAM para adicionar permissões do bucket do S3

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*`012345678901`*) pelo ID da conta.

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

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa a AWS Command Line Interface para criar e executar o aplicativo Flink do Managed Service for Apache Flink. Use o comando AWS CLI kinesisanalyticsv2 para criar e interagir com os aplicativos Managed Service para aplicativos Apache Flink.

Criação de uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação de ler no fluxo de origem, e outra que concede permissões para ações de gravação no fluxo de coleta. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua **`username`** pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos nomes do recurso da Amazon Resources Names (ARNs) **`(012345678901)`**pelo ID da sua conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

Criar um perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Perfis e Criar perfil.
3. Em Selecionar tipo de identidade de confiança, selecione AWSServiço.
4. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis.
5. Em Selecione seu caso de uso, selecione Managed Service for Apache Flink.
6. Selecione Next: Permissions (Próximo: permissões).
7. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
8. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.

9. Anexe a política de permissões à função.

> **Note**
>
> Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, Crie uma política de permissões.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).
   b. Selecione Attach Policies.
   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).
   d. Selecione a política `AKReadSourceStreamWriteSinkStream` e selecione Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo

Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (nome do usuário) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (012345678901) na função de execução do serviço pelo ID da conta.

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

Execute o CreateApplicationcom a seguinte solicitação para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação StartApplication para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

```
{{
    "ApplicationName": "s3_sink",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação StopApplication para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Execute a ação `StopApplication` com a solicitação anterior para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consulte Como configurar o registro de aplicativos.

Atualize as propriedades do ambiente

Nesta seção, você usa a ação UpdateApplication para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `update_properties_request.json`.

```
{"ApplicationName": "s3_sink",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "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"
               }
            }
         ]
      }
   }
}
```

2. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote de código, use a ação da UpdateApplicationCLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (<username>) com o sufixo que você selecionou na seção Criar recursos dependentes.

```
{
    "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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
```

```
            }
        }
    }
}
```

Limpar AWS recursos

Esta seção inclui procedimentos para limpar recursos AWS criados no tutorial Janela em cascata.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Exclua seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. no painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Exclua seus fluxos de dados do Kinesis

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Segurança no Amazon Managed Service for Apache Flink

A segurança na nuvem na AWS é a nossa maior prioridade. Como um cliente da AWS, você se beneficiará de um datacenter e uma arquitetura de rede criados para atender os requisitos da maioria das organizações com exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa produtos da AWS na Nuvem da AWS. A AWS também fornece serviços que podem ser usados com segurança. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de compatibilidade aplicáveis ao Managed Service for Apache Flink, consulte AWSServiços no escopo do programa de compatibilidade.
- Segurança da nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Managed Service for Apache Flink. Os tópicos a seguir mostram como configurar o Managed Service for Apache Flink para atender aos seus objetivos de segurança e de conformidade. Saiba também como usar outros serviços da Amazon que podem ajudar você a monitorar e proteger os recursos do Managed Service for Apache Flink.

Tópicos

Proteção de dados no Amazon Managed Service for Apache Flink

Você pode proteger seus dados usando ferramentas fornecidas pela AWS. O Managed Service for Apache Flink pode funcionar com serviços que oferecem suporte à criptografia de dados, incluindo o Kinesis Data Firehose e o Amazon S3.

Criptografia de dados no Managed Service for Apache Flink

Criptografia em repouso

Observe o seguinte sobre a criptografia de dados em repouso com o Managed Service for Apache Flink:

- Você pode criptografar dados no stream de dados de entrada do Kinesis usando o. StartStreamEncryption Para obter mais informações, consulte O que é criptografia do lado do servidor para o Kinesis Data Streams?.
- Os dados de saída podem ser criptografados em repouso usando o Kinesis Data Firehose para armazenar dados em um bucket criptografado do Amazon S3. É possível especificar a chave de criptografia que o bucket do Amazon S3 utiliza. Para obter mais informações, consulte Proteção de dados usando a criptografia do lado do servidor com chaves gerenciadas pelo KMS (SSE-KMS).
- O Managed Service para Apache Flink pode ler a partir de qualquer fonte de streaming e gravar em qualquer destino de streaming ou banco de dados. Garanta que suas fontes e destinos criptografem todos os dados em trânsito e em repouso.
- O código do aplicativo é criptografado em repouso.
- O armazenamento de aplicativos durável é criptografado em repouso.
- O armazenamento de aplicativos durável é criptografado em repouso.

Criptografia em trânsito

O Managed Service for Apache Flink criptografa todos os dados em trânsito. A criptografia em trânsito é habilitada para todos os aplicativos do Managed Service for Apache Flink e não pode ser desabilitada.

O Managed Service for Apache Flink criptografa todos os dados em trânsito.

- Dados em trânsito do Kinesis Data Streams para o Managed Service for Apache Flink.
- Dados em trânsito entre componentes internos no Managed Service for Apache Flink.

- Dados em trânsito do Kinesis Data Streams para o Managed Service for Apache Flink e Kinesis Data Firehose.

Gerenciamento de chaves

A criptografia de dados no Managed Service for Apache Flink usa chaves gerenciadas pelo serviço. Chaves gerenciadas pelo cliente não são compatíveis.

Gerenciamento de identidade e acesso para o Amazon Managed Service for Apache Flink

O AWS Identity and Access Management (IAM) é um AWS service (Serviço da AWS) que ajuda a controlar o acesso aos recursos da AWS de forma segura. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar os recursos do Managed Service for Apache Flink. O IAM é um AWS service (Serviço da AWS) que pode ser usado sem custo adicional.

Tópicos

Público

O uso do AWS Identity and Access Management (IAM) varia dependendo do trabalho que for realizado no Managed Service for Apache Flink.

Usuário do serviço – se você usa o Managed Service for Apache Flink para fazer o trabalho, o administrador fornece as credenciais e as permissões necessárias. À medida que você usar mais atributos do Managed Service for Apache Flink para fazer seu trabalho, você poderá precisar

de permissões adicionais. Entender como o acesso é gerenciado pode ajudá-lo a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um atributo no Managed Service for Apache Flink, consulte Gerenciamento de identidade e acesso para o Amazon Managed Service for Apache Flink.

Administrador do serviço – se você for o responsável pelos recursos do Managed Service for Apache Flink na empresa, provavelmente terá acesso total ao Managed Service for Apache Flink. É sua responsabilidade determinar quais recursos e atributos do Managed Service for Apache Flink os usuários do seu serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como a empresa pode usar o IAM com o Managed Service for Apache Flink, consulte Como o Amazon Managed Service for Apache Flink funciona com o IAM.

Administrador do IAM – se você for um administrador do IAM, talvez queira saber detalhes sobre como é possível criar políticas para gerenciar o acesso ao Managed Service for Apache Flink. Para visualizar exemplos de políticas baseadas em identidade do Managed Service for Apache Flink que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade do Amazon Managed Service for Apache Flink.

Autenticando com identidades

A autenticação é a forma como você faz login na AWS usando suas credenciais de identidade. É necessário ser autenticado (fazer login na AWS) como o usuário raiz da Usuário raiz da conta da AWS, como usuário do IAM ou assumindo um perfil do IAM.

É possível fazer login na AWS como uma identidade federada usando credenciais fornecidas por uma fonte de identidades. AWS IAM Identity Center Os usuários do IAM Identity Center, a autenticação única da empresa e as suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como uma identidade federada, o administrador já configurou anteriormente a federação de identidades utilizando perfis do IAM. Quando você acessa a AWS usando a federação, está indiretamente assumindo um perfil.

É possível fazer login no AWS Management Console ou no de acesso da AWS dependendo do tipo de usuário que você é. Para obter mais informações sobre como fazer login na AWS, consulte Conta da AWS Como fazer login na sua no Início de Sessão da AWS Guia do usuário.

Se você acessar a AWS programaticamente, a AWS fornecerá um kit de desenvolvimento de software (SDK) e uma interface de linha de comandos (CLI) para você assinar criptograficamente

as solicitações usando as suas credenciais. Se você não utilizar as ferramentas da AWS, deverá assinar as solicitações por conta própria. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinar solicitações de API da AWS no Guia do usuário do IAM.

Independentemente do método de autenticação usado, também pode ser exigido que você forneça mais informações de segurança. Por exemplo, a AWS recomenda o uso da autenticação multifator (MFA) para aumentar a segurança de sua conta. Para saber mais, consulte Autenticação multifator no Guia do usuário do AWS IAM Identity Center e Usar a autenticação multifator (MFA) naAWS no Guia do usuário do IAM.

Usuário raiz da Conta da AWS

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos os atributos e Serviços da AWS na conta. Essa identidade, denominada usuário raiz da Conta da AWS, e é acessada por login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não utilizar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários, inclusive os que precisam de acesso de administrador, usem a federação com um provedor de identidades para acessar Serviços da AWS usando credenciais temporárias.

Identidade federada é um usuário de seu diretório de usuários corporativos, um provedor de identidades da web AWS Directory Service, o , o diretório do Centro de Identidade ou qualquer usuário que acesse os Serviços da AWS usando credenciais fornecidas por meio de uma fonte de identidade. Quando as identidades federadas acessam Contas da AWS, elas assumem perfis que fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o .AWS IAM Identity Center É possível criar usuários e grupos no Centro de Identidade do IAM ou se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todas as suas Contas da AWS e aplicações. Para obter mais informações sobre o IAM Identity Center, consulte “O que é o IAM Identity Center?” no Guia do usuário do AWS IAM Identity Center.

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas para uma única pessoa ou aplicação. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de utilização específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível utilizar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, é possível ter um grupo chamado IAMAdmins e atribuir a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Um perfil do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. É possível assumir temporariamente um perfil do IAM no AWS Management Console alternando perfis. É possível assumir um perfil chamando uma operação de API da AWS CLI ou da AWS, ou usando um URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte Usar perfis do IAM no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar uma função para um provedor de identidade de terceiros no Guia do usuário do IAM. Se você usar o IAM Identity Center, deverá configurar um conjunto de permissões. Para controlar o que suas identidades podem acessar

após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário do AWS IAM Identity Center.

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, alguns Serviços da AWS permitem que você anexe uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Como os perfis do IAM diferem das políticas baseadas em recurso no Guia do usuário do IAM.
- Acesso entre serviços: alguns Serviços da AWS usam atributos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou uma função vinculada ao serviço.
  - Encaminhamento de sessões de acesso (FAS): qualquer pessoa que utilizar uma função ou usuário do IAM para realizar ações na AWS é considerada uma entidade principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O recurso FAS utiliza as permissões da entidade principal que chama um AWS service (Serviço da AWS), combinadas às permissões do AWS service (Serviço da AWS) solicitante, para realizar solicitações para serviços downstream. As solicitações de FAS só são feitas quando um serviço recebe uma solicitação que exige interações com outros Serviços da AWS ou com recursos para serem concluídas. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do usuário do IAM.
  - Perfil vinculado a serviço: um perfil vinculado a serviço é um tipo de perfil de serviço vinculado a um AWS service (Serviço da AWS). O serviço pode assumir o perfil de executar uma ação em seu nome. Os perfis vinculados ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

- Aplicações em execução no Amazon EC2: é possível usar um perfil do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir um perfil da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Usar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar os perfis do IAM, consulte Quando criar uma função do IAM (em vez de um usuário) no Guia do usuário do IAM.

Como gerenciar acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou recursos da AWS. Uma política é um objeto na AWS que, quando associado a uma identidade ou recurso, define suas permissões. A AWS avalia essas políticas quando uma entidade principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas são armazenadas na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar AWS as políticas JSON da para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM a perfis, e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de perfis do AWS Management Console, da AWS CLI ou da API da AWS.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas

políticas controlam quais ações os usuários e funções podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda mais como políticas embutidas ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários usuários, grupos e perfis na Conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recurso

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificada pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou Serviços da AWS.

Políticas baseadas em recursos são políticas em linha que estão localizadas nesse serviço. Não é possível usar as políticas gerenciadas da AWS do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços compatíveis com ACLs. Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

A AWS aceita tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs): SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (UO) no AWS Organizations. O AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS pertencentes à sua empresa. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades em contas-membro, incluindo cada .Usuário raiz da conta da AWS Para obter mais informações sobre o Organizações e SCPs, consulte How SCPs work (Como os SCPs funcionam) noAWS Organizations Guia do usuário do .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recurso. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir uma solicitação quando há vários tipos de política envolvidos, consulte Lógica da avaliaçãode políticas no Guia do usuário do IAM.

Como o Amazon Managed Service for Apache Flink funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Managed Service for Apache Flink, saiba quais recursos do IAM estão disponíveis para uso com o Managed Service for Apache Flink.

Recursos do IAM que você pode usar com o Amazon Managed Service for Apache Flink

atributo do IAM	Suporte do Amazon Managed Service for Apache Flink
Políticas baseadas em identidade	Sim
Políticas baseadas em recurso	Não
Ações de políticas	Sim
atributos de políticas	Sim
Chaves de condição de políticas	Não
ACLs	Não
ABAC (tags em políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Não
Perfis vinculados ao serviço	Não

Para obter uma visão geral de como o Managed Service for Apache Flink e outros serviços AWS funcionam com a maioria dos recursos do IAM, consulte Serviços AWS que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para o Managed Service for Apache Flink

Suporta políticas baseadas em identidade	Sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que é possível anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e funções podem realizar, em quais recursos e em que

condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou atributos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil à qual ela está anexado. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos da política JSON do IAM no Guia do Usuário do IAM.

Políticas baseadas em identidade para o Managed Service for Apache Flink

Para visualizar exemplos de políticas baseadas em identidade do Managed Service for Apache Flink, consulte Exemplos de políticas baseadas em identidade do Amazon Managed Service for Apache Flink.

Políticas baseadas em identidade para o Managed Service for Apache Flink

Oferece suporte a políticas baseadas em atributos	Sim

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificada pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou Serviços da AWS.

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recurso. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando a entidade principal e o recurso estão em diferentes Contas da AWS, um administrador do IAM da conta confiável também deve conceder à entidade principal (usuário ou perfil) permissão para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada

em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma outra política baseada em identidade será necessária. Para obter mais informações, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

Ações de políticas para o Managed Service for Apache Flink

Oferece suporte a ações de políticas	Sim

Os administradores podem usar as políticas de JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que é possível usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome que a operação de API da AWS associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista das ações do Managed Service for Apache Flink, consulte Ações definidas pelo Amazon Managed Service for Apache Flink na Referência de autorização de serviço.

As ações de políticas no Managed Service for Apache Flink usam o seguinte prefixo antes da ação:

```
Kinesis Analytics
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "Kinesis Analytics:action1",
      "Kinesis Analytics:action2"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "Kinesis Analytics:Describe*"
```

Para visualizar exemplos de políticas baseadas em identidade do Managed Service for Apache Flink, consulte Exemplos de políticas baseadas em identidade do Amazon Managed Service for Apache Flink.

Recursos de políticas para o Managed Service for Apache Flink

Oferece suporte a atributos de políticas	Sim

Os administradores podem usar as políticas de JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política `Resource` JSON especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou um elemento `NotResource`. Como prática recomendada, especifique um recurso usando seu Nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem suporte a um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem suporte a permissões em nível de recurso, como operações de listagem, use um caractere curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Managed Service for Apache Flink e seus ARNs, consulte Recursos definidos pelo Amazon Managed Service for Apache Flink na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Managed Service for Apache Flink.

Para visualizar exemplos de políticas baseadas em identidade do Managed Service for Apache Flink, consulte Exemplos de políticas baseadas em identidade do Amazon Managed Service for Apache Flink.

Chaves de condições de políticas para o Managed Service for Apache Flink

Suporta chaves de condição de política específicas de serviço	Sim

Os administradores podem usar as políticas de JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` (ou `Condition` bloco de) permite que você especifique condições nas quais uma instrução está em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usam atendentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, a AWS avaliará a condição usando uma operação lógica `OR`. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

É possível também usar variáveis de espaço reservado ao especificar as condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um recurso somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos de política do IAM: variáveis e tags no Guia do usuário do IAM.

A AWS oferece suporte a chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as AWS chaves de condição globais da , consulte AWSChaves de contexto de condição globais da no Guia do usuário do IAM.

Para ver uma lista das chaves de condição do Managed Service for Apache Flink, consulte Chaves de condição para o Amazon Managed Service for Apache Flink na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Managed Service for Apache Flink.

Para visualizar exemplos de políticas baseadas em identidade do Managed Service for Apache Flink, consulte Exemplos de políticas baseadas em identidade do Amazon Managed Service for Apache Flink.

Listas de controle de acesso (ACLs) no Managed Service for Apache Flink

Oferece suporte a ACLs	Não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Controle de acesso por atributo (ABAC) com o Managed Service for Apache Flink

Oferece suporte a ABAC (tags em políticas)	Sim

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Na AWS, esses atributos são chamados de tags. É possível anexar tags a entidades do IAM (usuários ou perfis) e a muitos recursos da AWS. A marcação de entidades e atributos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do atributo que ela está tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações onde o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys` chaves de condição.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial.

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

Usando credenciais temporárias com o Managed Service for Apache Flink

Oferece suporte a credenciais temporárias	Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Você está usando credenciais temporárias se faz login no AWS Management Console usando qualquer método, exceto um nome de usuário e uma senha. Por exemplo, quando você

acessa a AWS usando o link de autenticação única (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar perfis, consulte Alternar para um perfil (console) no Guia do usuário do IAM.

É possível criar credenciais temporárias manualmente usando a AWS CLI ou a API da AWS. Em seguida, você pode usar essas credenciais temporárias para acessar aAWS .AWS A recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias no IAM.

Permissões de entidade principal entre serviços para o Managed Service for Apache Flink

Suporte para o recurso Encaminhamento de sessões de acesso (FAS)	Sim

Quando você usa um usuário ou um perfil do IAM para executar ações na AWS, você é considerado uma entidade principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O recurso FAS utiliza as permissões da entidade principal que chama um AWS service (Serviço da AWS), combinadas às permissões do AWS service (Serviço da AWS) solicitante, para realizar solicitações para serviços downstream. As solicitações de FAS só são feitas quando um serviço recebe uma solicitação que exige interações com outros Serviços da AWS ou com recursos para serem concluídas. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

Perfis de serviço para o Managed Service for Apache Flink

Oferece suporte a perfis de serviço	Sim

O perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do usuário do IAM.

> **Warning**
>
> Alterar as permissões de um perfil de serviço pode prejudicar a funcionalidade do Managed Service for Apache Flink. Só edite os perfis de serviço quando o Managed Service for Apache Flink orientar você a fazê-lo.

Perfis vinculadas ao serviço do Managed Service for Apache Flink

Oferece suporte a perfis vinculados ao serviço	Sim

Um perfil vinculado ao serviço é um tipo de perfil de serviço vinculado a um AWS service (Serviço da AWS). O serviço pode assumir o perfil de executar uma ação em seu nome. Os perfis vinculados ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar funções vinculadas a serviços, consulte Serviços do AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Selecione o link Sim para visualizar a documentação da função vinculada a serviço desse serviço.

Exemplos de políticas baseadas em identidade do Amazon Managed Service for Apache Flink

Por padrão, os usuários e os perfis não têm permissão para criar ou modificar os recursos do Managed Service for Apache Flink. Eles também não podem executar tarefas usando o AWS Management Console, a AWS Command Line Interface (AWS CLI) ou a API AWS. Para conceder aos usuários permissão para executar ações nos recursos de que precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis, e os usuários podem assumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do Usuário do IAM.

Para obter detalhes sobre ações e tipos de recurso definidos pelo Managed Service for Apache Flink, incluindo o formato dos ARNs para cada tipo de recurso, consulte Ações, recursos e chaves de condição do Amazon Managed Service for Apache Flink na Referência de autorização de serviço.

Tópicos

Práticas recomendadas de políticas

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Managed Service for Apache Flink em sua conta. Essas ações podem incorrer em custos para a Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com AWS as políticas gerenciadas pela e avance para as permissões de privilégio mínimo: para começar a conceder permissões a seus usuários e workloads, use as AWS políticas gerenciadas pela que concedem permissões para muitos casos de uso comuns. Elas estão disponíveis na sua Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da AWS específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. É possível também usar condições para conceder acesso a ações de serviço, se elas forem usadas por meio de um AWS service (Serviço da AWS) específico, como o AWS CloudFormation. Para obter mais informações, consulte Elementos de política JSON do IAM: Condição no Manual do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudá-lo

a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do usuário do IAM.

- Exigir autenticação multifator (MFA): se houver um cenário que exija usuários do IAM ou um usuário raiz em sua Conta da AWS, ative a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usando o console do Managed Service for Apache Flink

Para acessar o console do Amazon Managed Service for Apache Flink, você deve ter um conjunto mínimo de permissões. Essas permissões devem autorizar você a listar e visualizar detalhes sobre os recursos do Managed Service for Apache Flink na sua Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Não é necessário conceder permissões mínimas do console para usuários que fazem chamadas somente à AWS CLI ou à AWS API. Em vez disso, permita o acesso somente a ações que correspondam a operação de API que estiverem tentando executar.

Para garantir que os usuários e os perfis ainda possam usar o console do Managed Service for Apache Flink, anexe também a política gerenciada `ConsoleAccess` ou `ReadOnly` AWS do Managed Service for Apache Flink às entidades. Para obter mais informações, consulte Adicionando Permissões a um Usuário no Guia do Usuário do IAM.

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como é possível criar uma política que permite que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a AWS API.

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

```
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Gerenciamento de identidade e acesso para o Amazon Managed Service for Apache Flink

Use as seguintes informações para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao trabalhar com o Managed Service for Apache Flink e o IAM.

Tópicos

- Não tenho autorização para executar uma ação no Managed Service for Apache Flink
- Não estou autorizado a realizar iam: PassRole

- Desejo permitir que pessoas fora da minha conta AWS acessem meus recursos do Managed Service for Apache Flink

Não tenho autorização para executar uma ação no Managed Service for Apache Flink

Se o AWS Management Console informar que você não foi autorizado a executar uma ação, você deve entrar em contato com o administrador para obter assistência. O administrador é a pessoa que forneceu o seu nome de usuário e senha.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `Kinesis Analytics:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
 Analytics:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas e permitir o acesso ao recurso *`my-example-widget`* usando a ação `Kinesis Analytics:`*`GetWidget`*.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não tem autorização para executar a ação `iam:PassRole`, as suas políticas deverão ser atualizadas para permitir a passagem de um perfil para o Managed Service for Apache Flink.

Alguns Serviços da AWS permitem que você passe um perfil existente para o serviço, em vez de criar um novo perfil de serviço ou perfil vinculado ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro de exemplo a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Managed Service for Apache Flink. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

Desejo permitir que pessoas fora da minha conta AWS acessem meus recursos do Managed Service for Apache Flink

É possível criar uma função que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir a função. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se um Managed Service for Apache Flink oferece suporte a esses recursos, consulte Como o Amazon Managed Service for Apache Flink funciona com o IAM.
- Para saber como conceder acesso a seus atributos em todas as Contas da AWS pertencentes a você, consulte Fornecimento de acesso a um usuário do IAM em outra Conta da AWS pertencente a você no Guia de usuário do IAM.
- Para saber como conceder acesso a seus recursos para terceiros Contas da AWS, consulte Fornecimento de acesso a Contas da AWS pertencentes a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em recurso para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em recurso no Guia do usuário do IAM.

Prevenção do problema “confused deputy” entre serviços

Em AWS, a personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a atuar nos recursos de outro cliente, mesmo sendo que ele não deveria ter as permissões adequadas, resultando em um problema de “confused deputy”.

Para evitar confused deputies, o AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços que usam entidades principais de serviço que receberam acesso aos

recursos em sua conta. Esta seção se concentra na prevenção de confused deputies entre serviços, específica ao Managed Service for Apache Flink. No entanto, você pode saber mais sobre esse tópico na seção O problema do "confused deputy" do Guia do usuário do IAM.

No contexto do Managed Service for Apache Flink, recomendamos usar as chaves de contexto de condição SourceAccount global aws: SourceArn e aws: na política de confiança de sua função para limitar o acesso à função somente às solicitações geradas pelos recursos esperados.

Use `aws:SourceArn` se quiser que apenas um recurso seja associado ao acesso entre serviços. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

O valor de `aws:SourceArn` deve ser o ARN do recurso usado pelo Managed Service for Apache Flink, que é especificado com o seguinte formato: `arn:aws:kinesisanalytics:region:account:resource`.

A abordagem recomendada para o problema do confused deputy é usar a chave de contexto de condição global `aws:SourceArn` com o ARN com recursos completos.

Se você não souber o ARN completo do recurso ou estiver especificando vários recursos, use a chave `aws:SourceArn` com caracteres curingas (*) para as partes desconhecidas do ARN. Por exemplo: `arn:aws:kinesisanalytics::111122223333:*`.

As políticas de perfis que você fornece ao Managed Service for Apache Flink, assim como as políticas de confiança dos perfis gerados para você, podem fazer uso dessas chaves.

Para se proteger contra o problema do confused deputy, execute as seguintes tarefas:

Para se proteger contra o problema do confused deputy

1. Faça login no Console de Gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Selecione Perfis e, em seguida, selecione o perfil que você deseja modificar.
3. Selecione Edit trust policy (Editar política de confiança).
4. Na página Editar política de confiança, substitua a política JSON padrão por uma política que usa uma ou ambas as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount`. Veja os exemplos de políticas a seguir:
5. Selecione Update policy.

```
{
```

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

Monitorando o Amazon Managed Service for Apache Flink

O Managed Service for Apache Flink fornece uma funcionalidade de monitoramento para seus aplicativos. Para ter mais informações, consulte Registro e Monitoramento.

Validação de conformidade Amazon Managed Service for Apache Flink

Auditores externos avaliam a segurança e a conformidade do Amazon Managed Service for Apache Flink como parte de vários programas de compatibilidade AWS. Isso inclui SOC, PCI, HIPAA e outros.

Para obter uma lista dos serviços AWS que estão no escopo de programas de conformidade específicos, consulte . Para obter informações gerais, consulte Programas de conformidade da AWS.

É possível baixar relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Fazer download dos relatórios no AWS Artifact.

Sua responsabilidade de conformidade ao usar o Managed Service for Apache Flink é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelos regulamentos e leis aplicáveis. Se seu uso do Managed Service for Apache Flink estiver sujeito à conformidade com padrões como HIPAA ou PCI, AWS fornecerá recursos para ajudar:

- Guias de início rápido de segurança e conformidade – Esses guias de implantação discutem considerações sobre arquitetura e fornecem medidas para implantar ambientes de linha de base focados em segurança e conformidade na AWS.
- Architecting for HIPAA Security and Compliance on Amazon Web Services. Este whitepaper descreve como as empresas podem usar a AWS para criar aplicativos em conformidade com a HIPAA.
- Recursos de compatibilidade da AWS – Esta coleção de guias e pastas de trabalho pode ser aplicada ao seu setor e local.
- AWS Config: esse serviço da AWS avalia até que ponto suas configurações de recursos atendem adequadamente às práticas internas e às diretrizes e regulamentações do setor.
- AWS Security Hub: esse serviço da AWS fornece uma visão abrangente do estado da segurança na AWS que ajuda verificar a conformidade com os padrões e as práticas recomendadas de segurança do setor.

FedRAMP

O programa de compatibilidade do FedRAMP AWS inclui o Managed Service for Apache Flink como um serviço autorizado pelo FedRAMP. Se você for um cliente federal ou comercial, poderá usar o serviço para processar e armazenar cargas de trabalho confidenciais no limite de autorização da região AWS GovCloud (EUA) com dados até o nível de alto impacto, bem como nas regiões Leste dos EUA (Norte da Virgínia), Leste dos EUA (Ohio), Oeste dos EUA (Norte da Califórnia) e Oeste dos EUA (Oregon) com dados até um nível moderado.

Você pode solicitar acesso aos pacotes de segurança do FedRAMP da AWS diretamente com o Project Management Office (PMO – Escritório de gerenciamento de projetos) do FedRAMP ou com o seu gerente de conta da área de vendas da AWS, ou baixá-los no AWS Artifact em AWSArtifact.

Para obter mais informações, consulte FedRAMP.

Resiliência no Amazon Managed Service for Apache Flink

A infraestrutura global da AWS é criada com base em regiões da AWS e zonas de disponibilidade da AWS. As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, as quais são conectadas com baixa latência, alto throughput e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte AWS Infraestrutura Global.

Além da infraestrutura global da AWS, um Managed Service for Apache Flink oferece vários recursos para ajudar a oferecer suporte às suas necessidades de resiliência de dados e backup.

Recuperação de desastres

O Managed Service for Apache Flink é executado em um modo sem servidor e cuida das degradações do host, da disponibilidade da zona de disponibilidade e outros problemas relacionados à infraestrutura, fazendo uma migração automática. O Managed Service for Apache Flink consegue isso por meio de vários mecanismos redundantes. Cada aplicativo Managed Service for Apache Flink é executado em um cluster do Apache Flink com locatário único. O cluster Apache Flink é executado no modo de alta disponibilidade usando o Zookeeper JobMananger em várias zonas de disponibilidade. O Managed Service for Apache Flink implanta o Apache Flink usando o Amazon EKS. Vários pods do Kubernetes são usados no Amazon EKS para cada região AWS em todas as zonas de disponibilidade. No caso de uma falha, o Managed Service for Apache Flink tenta em primeiro lugar recuperar o aplicativo dentro do cluster do Apache Flink em execução usando os pontos de verificação do seu aplicativo, se disponível.

O Managed Service for Apache Flink faz backup do estado do aplicativo usando pontos de verificação e snapshots:

- Os pontos de verificação são backups do estado do aplicativo que o Managed Service for Apache Flink cria automaticamente e periodicamente e para restaurar falhas.
- Os Snapshots são backups do estado do aplicativo que você cria e restaura manualmente.

Para obter mais informações sobre os pontos de verificação e os snapshots, consulte Tolerância a falhas.

Versionamento

As versões armazenadas do estado do aplicativo são versionadas da seguinte forma:

- Os pontos de verificação são versionados automaticamente pelo serviço. Se o serviço usar um ponto de verificação para reiniciar o aplicativo, o ponto de verificação mais recente será usado.
- Os pontos de salvamento são versionados usando o SnapshotNameparâmetro da ação. CreateApplicationSnapshot

O Managed Service for Apache Flink criptografa dados armazenados em pontos de verificação e salvamento.

Segurança da infraestrutura no Managed Service for Apache Flink

Como um serviço gerenciado, o Managed Service for Apache Flink é protegido pelos procedimentos de segurança de rede global da AWS que estão descritos no whitepaper Amazon Web Services: visão geral dos processos de segurança.

Você usa chamadas de API publicadas pela AWS para acessar o Managed Service for Apache Flink por meio da rede. Todas as chamadas de API para o Managed Service for Apache Flink são protegidas via Transport Layer Security (TLS) e autenticadas via IAM. Os clientes devem ser compatíveis com o TLS 1.2 ou posterior. Os clientes também devem ter suporte a conjuntos de criptografia com perfect forward secrecy (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece suporte a esses modos.

Além disso, as solicitações devem ser assinadas utilizando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Melhores práticas de segurança para o Managed Service for Apache Flink

O Amazon Managed Service for Apache Flink fornece uma série de recursos de segurança a serem considerados no desenvolvimento e na implementação das suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes no seu ambiente, trate-as como considerações úteis em vez de requisitos.

Implemente o acesso de privilégio mínimo

Ao conceder permissões, você decide quem receberá quais permissões para quais recursos do Managed Service for Apache Flink. Você habilita ações específicas que quer permitir nesses recursos. Portanto, você deve conceder somente as permissões necessárias para executar uma tarefa. A implementação do privilégio de acesso mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

Use perfis do IAM para acessar outros serviços da Amazon

O aplicativo do Managed Service for Apache Flink deve ter credenciais válidas para acessar recursos em outros serviços, como fluxos de dados do Kinesis, fluxos do Kinesis Data Firehose ou buckets do Amazon S3. Você não deve armazenar credenciais da AWS diretamente no aplicativo ou em um bucket do Amazon S3. Essas são credenciais de longo prazo que não são automaticamente alternadas e podem ter um impacto comercial significativo se forem comprometidas.

Em vez disso, você deve usar um perfil do IAM para gerenciar credenciais temporárias para que o aplicativo acesse outros recursos. Quando você usa um perfil, não precisa usar credenciais de longo prazo para acessar outros recursos.

Para obter mais informações, consulte os seguintes tópicos no Manual do usuário do IAM:

- Funções do IAM
- Cenários comuns para funções: usuários, aplicativos e serviços

Implemente a criptografia do lado do servidor em recursos dependentes

Dados em repouso e dados em trânsito são criptografados no Managed Service for Apache Flink e essa criptografia não pode ser desabilitada. É necessário implementar a criptografia do lado

do servidor em seus recursos dependentes, como fluxos de dados do Kinesis, fluxos do Kinesis Data Firehose e buckets do Amazon S3. Para obter mais informações sobre como implementar a criptografia do lado do servidor em recursos dependentes, consulte Proteção de dados.

Use CloudTrail para monitorar chamadas de API

O Managed Service for Apache Flink é integrado ao AWS CloudTrail, um serviço que fornece um registro das ações executadas por um usuário, perfil ou serviço da Amazon no Managed Service for Apache Flink.

Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Managed Service for Apache Flink, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para ter mais informações, consulte the section called “Usando o AWS CloudTrail”.

Log e monitoramento no Amazon Managed Service for Apache Flink

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho dos aplicativos Managed Service for Apache Flink. Você deve coletar dados de monitoramento de todas as partes da solução da AWS para facilitar a depuração de uma falha de vários pontos, caso ocorra.

Antes de começar a monitorar o Managed Service for Apache Flink, crie um plano de monitoramento que inclua as respostas para as seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer um parâmetro de desempenho normal do Managed Service for Apache Flink em seu ambiente. Você faz isso medindo o desempenho em vários momentos e em condições diferentes de carga. Enquanto monitora o Managed Service for Apache Flink, você pode armazenar dados históricos de monitoramento. Você poderá, em seguida, compará-los com os dados de desempenho atuais, identificar padrões de desempenho normais e anomalias de desempenho, e elaborar métodos para resolver problemas.

Tópicos

Registro em log

O registro em log é importante para que os aplicativos de produção entendam erros e falhas. No entanto, o subsistema de registro precisa coletar e encaminhar entradas de registro para CloudWatch registros. Embora alguns registros sejam bons e desejáveis, um registro extensivo pode sobrecarregar o serviço e fazer com que o aplicativo Flink fique para trás. O log de exceções e avisos certamente é uma boa ideia. Mas, você não pode gerar uma mensagem de log para cada mensagem processada pelo aplicativo Flink. O Flink é otimizado para latências altas constantes e baixas, mas o subsistema de registro não é. Caso seja realmente necessário gerar uma saída de log para cada mensagem processada, use um coletor adicional DataStream dentro do aplicativo Flink e um coletor adequado para enviar os dados para o Amazon CloudWatch S3 ou. Não use o sistema de log Java para essa finalidade. Além disso, a `Debug Monitoring Log Level` configuração do Managed Service for Apache Flink gera uma grande quantidade de tráfego, o que pode criar contrapressão. Você só deve usá-lo enquanto estiver investigando ativamente os problemas com o aplicativo.

Consultando registros com o Logs CloudWatch Insights

CloudWatch O Logs Insights é um serviço poderoso para consultar registros em grande escala. Os clientes devem aproveitar seus recursos para pesquisar rapidamente os logs para identificar e mitigar erros durante eventos operacionais.

A consulta a seguir procura exceções em todos os registros do gerenciador de tarefas e as ordena de acordo com a hora em que ocorreram.

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
 @message like /(Error|Exception)/
| sort @timestamp desc
```

Para outras consultas úteis, consulte Exemplos de consultas.

Monitoramento

Ao executar aplicativos de streaming em produção, você decide executar o aplicativo de forma contínua e indefinida. É fundamental implementar o monitoramento e o alarme adequados para todos os componentes, não apenas os do aplicativo Flink. Caso contrário, você corre o risco de deixar

passar problemas emergentes logo no início e só perceber um evento operacional quando ele estiver totalmente elucidado e for muito mais difícil de mitigar. No geral é preciso monitorar:

- A fonte está ingerindo dados?
- Os dados são lidos a partir da fonte (da perspectiva da fonte)?
- O aplicativo Flink está recebendo dados?
- O aplicativo Flink é capaz de acompanhar ou está ficando para trás?
- O aplicativo Flink está persistindo em colocar dados no coletor (do ponto de vista do aplicativo)?
- O coletor está recebendo dados?

Métricas mais específicas devem, em seguida, ser consideradas para o aplicativo Flink. Esse CloudWatch painel fornece um bom ponto de partida. Para obter mais informações sobre quais métricas monitorar para aplicativos de produção, consulte Usando CloudWatch alarmes com o Amazon Managed Service para Apache Flink. Essas métricas incluem:

- records_lag_max e MillisBehindLatest – se o aplicativo estiver consumindo do Kinesis ou do Kafka, estas métricas indicam se o aplicativo está ficando para trás e precisa ser agilizado para acompanhar a carga atual. Essa é uma boa métrica genérica que é fácil de rastrear para todos os tipos de aplicativos. Mas, ele só pode ser usado para escalonamento reativo, ou seja, quando o aplicativo já ficou para trás.
- CPUUtilization heapMemoryUtilizatione — Essas métricas fornecem uma boa indicação da utilização geral dos recursos do aplicativo e podem ser usadas para escalabilidade proativa, a menos que o aplicativo esteja vinculado à E/S.
- Tempo de inatividade – um tempo de inatividade maior que zero indica que o aplicativo falhou. Se o valor for maior que zero, o aplicativo não está processando nenhum dado.
- lastCheckpointSizee lastCheckpointDuration— Essas métricas monitoram a quantidade de dados armazenados no estado e quanto tempo é necessário para chegar a um ponto de verificação. Se os pontos de verificação aumentarem ou demorarem muito, o aplicativo gastará tempo continuamente fazendo pontos de verificação e terá menos ciclos para o processamento real. Em alguns pontos, os pontos de verificação podem ficar muito grandes ou demorar tanto tempo que acabam falhando. Além de monitorar valores absolutos, os clientes também devem considerar monitorar a taxa de alteração com `RATE(lastCheckpointSize)` e `RATE(lastCheckpointDuration)`.
- numberOfFailedPontos de verificação — Essa métrica conta o número de pontos de verificação com falha desde o início do aplicativo. Dependendo do aplicativo, pode ser tolerável que os pontos

de verificação falhem de vez em quando. Mas, se os pontos de verificação falharem regularmente, é provável que o aplicativo não esteja íntegro e precise de mais atenção. Recomendamos o monitoramento `RATE(numberOfFailedCheckpoints)` para gerar alertas sobre o gradiente e não sobre os valores absolutos.

Configurando o log de aplicativos

Ao adicionar uma opção de CloudWatch registro da Amazon ao seu aplicativo Managed Service for Apache Flink, você pode monitorar eventos do aplicativo ou problemas de configuração.

Este tópico descreve como configurar seu aplicativo para gravar eventos do aplicativo em um stream do CloudWatch Logs. Uma opção de CloudWatch registro é uma coleção de configurações e permissões do aplicativo que seu aplicativo usa para configurar a forma como grava eventos do aplicativo no CloudWatch Logs. Você pode adicionar e configurar uma opção de CloudWatch registro usando o AWS Management Console ou o AWS Command Line Interface (AWS CLI).

Observe o seguinte sobre como adicionar uma opção de CloudWatch registro ao seu aplicativo:

- Quando você adiciona uma opção de CloudWatch registro usando o console, o Managed Service for Apache Flink cria o grupo de CloudWatch log e o stream de log para você e adiciona as permissões que seu aplicativo precisa para gravar no stream de log.
- Ao adicionar uma opção de CloudWatch registro usando a API, você também deve criar o grupo de registros e o fluxo de registros do aplicativo e adicionar as permissões de que seu aplicativo precisa para gravar no fluxo de registros.

Este tópico contém as seguintes seções:

Configurando o CloudWatch registro usando o console

Quando você ativa o CloudWatch registro em log para seu aplicativo no console, um grupo de CloudWatch registros e um fluxo de registros são criados para você. Além disso, a política de permissões do seu aplicativo é atualizada com permissões para gravar no fluxo.

O Managed Service for Apache Flink cria um grupo de logs chamado usando a seguinte convenção, onde *ApplicationName* está o nome do seu aplicativo.

```
/AWS/KinesisAnalytics/ApplicationName
```

O Managed Service for Apache Flink cria um fluxo de logs no novo grupo de logs com o nome a seguir.

```
kinesis-analytics-log-stream
```

Você define o nível da métrica de monitoramento do aplicativo e o nível do log de monitoramento usando a seção Monitoring log level (Nível do log de monitoramento) da página Configure application (Configurar aplicativo). Para obter informações sobre os níveis de log do aplicativo, consulte the section called “Níveis de monitoramento de aplicativos”.

Configurando o CloudWatch registro usando a CLI

Para adicionar uma opção de CloudWatch registro usando oAWS CLI, faça o seguinte:

- Crie um grupo de CloudWatch registros e um fluxo de registros.
- Adicione uma opção de registro ao criar um aplicativo usando a CreateApplicationação ou adicione uma opção de registro a um aplicativo existente usando a AddApplicationCloudWatchLoggingOptionação.
- Adicione permissões à política do seu aplicativo para gravar nos logs.

Esta seção contém os seguintes tópicos:

- Criando um grupo de CloudWatch registros e um fluxo de registros
- Trabalhando com opções de CloudWatch registro de aplicativos
- Adicionando permissões para gravação no fluxo de CloudWatch registros

Criando um grupo de CloudWatch registros e um fluxo de registros

Você cria um grupo de CloudWatch registros e transmite usando o console de CloudWatch registros ou a API. Para obter informações sobre como criar um grupo de CloudWatch registros e um fluxo de registros, consulte Como trabalhar com grupos de registros e fluxos de registros.

Trabalhando com opções de CloudWatch registro de aplicativos

Use as ações de API a seguir para adicionar uma opção de CloudWatch log a um aplicativo novo ou existente ou alterar uma opção de log de um aplicativo existente. Para obter informações sobre como usar um arquivo JSON como entrada de uma ação da API, consulte Exemplo de código de API para o Managed Service for Apache Flink.

Adicionando uma opção de CloudWatch registro ao criar um aplicativo

O exemplo a seguir demonstra como usar a `CreateApplication` ação para adicionar uma opção de CloudWatch log ao criar um aplicativo. No exemplo, substitua o *Amazon Resource Name (ARN) do stream de CloudWatch log para adicioná-lo ao novo aplicativo* com suas próprias informações. Para obter mais informações sobre a ação, 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>"
 }]
```

```
}
```

Adicionando uma opção de CloudWatch log a um aplicativo existente

O exemplo a seguir demonstra como usar a `AddApplicationCloudWatchLoggingOption` ação para adicionar uma opção de CloudWatch log a um aplicativo existente. No exemplo, substitua cada *`espaço reservado para entrada do usuário`* por suas próprias informações. Para obter mais informações sobre a ação, 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>
}
```

Atualizando uma opção de CloudWatch registro existente

O exemplo a seguir demonstra como usar a `UpdateApplication` ação para modificar uma opção de CloudWatch log existente. No exemplo, substitua cada *`espaço reservado para entrada do usuário`* por suas próprias informações. Para obter mais informações sobre a ação, 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>
}
```

Excluindo uma opção de CloudWatch log de um aplicativo

O exemplo a seguir demonstra como usar a `DeleteApplicationCloudWatchLoggingOption` ação para excluir uma opção de CloudWatch log existente. No exemplo, substitua cada *`espaço`*

reservado para entrada do usuário por suas próprias informações. Para obter mais informações sobre a ação, 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>
}
```

Definindo o nível de registro em log do aplicativo

Para definir o nível de registro em log do aplicativo, use o parâmetro MonitoringConfiguration da ação CreateApplication ou o parâmetro MonitoringConfigurationUpdate da ação UpdateApplication.

Para obter informações sobre os níveis de log do aplicativo, consulte the section called “Níveis de monitoramento de aplicativos”.

Defina o nível de registro em log do aplicativo ao criar um aplicativo

O exemplo de solicitação a seguir para a ação CreateApplication define o nível de log do aplicativo como 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"
}
```

Atualize o nível de registro em log do aplicativo

O exemplo de solicitação a seguir para a ação `UpdateApplication` define o nível de log do aplicativo como `INFO`.

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "LogLevelUpdate": "INFO"
         }
      }
   }
}
```

Adicionando permissões para gravação no fluxo de CloudWatch registros

O Managed Service for Apache Flink precisa de permissões para gravar erros de configuração incorreta. CloudWatch Você pode adicionar essas permissões ao perfil do (IAM) AWS Identity and Access Management que o Managed Service for Apache Flink assumir.

Para obter mais informações sobre como usar um perfil do IAM para o Managed Service for Apache Flink, consulte Gerenciamento de identidade e acesso para o Amazon Managed Service for Apache Flink.

Política de confiança

Para conceder permissões ao Managed Service for Apache Flink para assumir o perfil do IAM, anexe a política de confiança a seguir ao perfil de execução de serviço.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanlaytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Política de permissões

Para conceder permissões a um aplicativo para gravar eventos de log a CloudWatch partir de um recurso do Managed Service for Apache Flink, você pode usar a seguinte política de permissões do IAM. Forneça os nomes do recurso da Amazon (ARN) corretos para seu grupo e seu fluxo de logs.

```
{
    "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:*",
            ]
        }
    ]
}
```

Níveis de monitoramento de aplicativos

Você controla a geração de mensagens de log do aplicativo usando o Nível de métricas de monitoramento e o Nível de registro em log de monitoramento do aplicativo.

O nível das métricas de monitoramento do aplicativo controla a granularidade das mensagens de log. Os níveis de métricas de monitoramento são definidos da seguinte forma:

- Aplicativo: as métricas têm como escopo o aplicativo todo.
- Tarefa: as métricas têm como escopo cada tarefa. Para obter mais informações sobre as tarefas, consulte the section called “Escalabilidade”.
- Operador: as métricas têm como escopo cada operador. Para obter mais informações sobre os operadores, consulte the section called “Operadores da API do DataStream”.
- Paralelismo: as métricas têm como escopo o paralelismo de aplicativos. Você só pode definir esse nível de métrica usando o MonitoringConfigurationUpdateparâmetro da UpdateApplicationAPI. Você não pode definir esse nível de métricas usando o console. Para obter mais informações sobre paralelismo, consulte the section called “Escalabilidade”.

O nível do registro em log de monitoramento do aplicativo controla a verbosidade do log do aplicativo. Os níveis do registro em log de monitoramento são definidos da seguinte forma:

- Erro possíveis eventos catastróficos do aplicativo.
- Alerta: situações potencialmente prejudiciais do aplicativo.
- Info: eventos de falha informativos e transitórios do aplicativo. Recomendamos usar esse nível de log.
- Depuração: eventos informativos detalhados que são mais úteis para depurar um aplicativo. Observação: use esse nível somente para fins de depuração temporária.

Práticas recomendadas de registro em log

Recomendamos que seu aplicativo use o nível Info de registro em log. Recomendamos esse nível para garantir que você veja os erros do Apache Flink, que são registrados em log no nível Info e não no nível Error.

Recomendamos que você use o nível de Debug apenas temporariamente ao investigar problemas do aplicativo. Volte para o nível Info quando o problema for resolvido. Usar o nível Debug de registro em log afetará significativamente o desempenho do seu aplicativo.

O registro em log excessivo também pode afetar significativamente o desempenho do aplicativo. Recomendamos que você não grave log para cada registro processado, por exemplo. O registro

excessivo pode causar gargalos graves no processamento de dados e causar contrapressão na leitura de dados das fontes.

Solução de problemas do registro em log

Se os registros do aplicativo não estiverem sendo gravados no fluxo de logs, verifique o seguinte:

- Se o perfil e as políticas do IAM do seu aplicativo estão corretos. Se a política do seu aplicativo precisa das seguintes permissões para acessar seu fluxo de logs:
  - `logs:PutLogEvents`
  - `logs:DescribeLogGroups`
  - `logs:DescribeLogStreams`

  Para ter mais informações, consulte the section called “Adicionando permissões para gravação no fluxo de CloudWatch registros”.
- Verifique se o seu aplicativo está sendo executado. Para verificar o status do seu aplicativo, visualize a página do seu aplicativo no console ou use as ListApplicationsações DescribeApplicationou.
- Monitore CloudWatch métricas como `downtime` para diagnosticar outros problemas do aplicativo. Para obter informações sobre CloudWatch métricas de leitura, consulteMétricas e dimensões no Managed Service for Apache Flink.

Próxima etapa

Depois de habilitar o CloudWatch login no seu aplicativo, você pode usar o CloudWatch Logs Insights para analisar os registros do seu aplicativo. Para ter mais informações, consulte the section called “Analisando logs”.

Análise de registros com o CloudWatch Logs Insights

Depois de adicionar uma opção de CloudWatch registro ao seu aplicativo, conforme descrito na seção anterior, você pode usar o CloudWatch Logs Insights para consultar seus fluxos de registros em busca de eventos ou erros específicos.

CloudWatch O Logs Insights permite que você pesquise e analise interativamente seus dados de registro no CloudWatch Logs.

Para obter informações sobre como começar a usar o CloudWatch Logs Insights, consulte Analisar dados de registro com o CloudWatch Logs Insights.

Executar uma consulta de amostra

Esta seção descreve como executar um exemplo de consulta do CloudWatch Logs Insights.

Pré-requisitos

- Grupos de registros e fluxos de registros existentes configurados no CloudWatch Logs.
- Registros existentes armazenados em CloudWatch Registros.

Se você usa serviços como AWS CloudTrail o Amazon Route 53 ou o Amazon VPC, provavelmente já configurou os registros desses serviços para CloudWatch acessar o Logs. Para mais informações sobre o envio de CloudWatch registros para o Logs, consulte Introdução aos CloudWatch registros.

As consultas no CloudWatch Logs Insights retornam um conjunto de campos de eventos de log ou o resultado de uma agregação matemática ou outra operação realizada em eventos de log. Esta seção demonstra uma consulta que retorna uma lista de eventos de log.

Para executar uma consulta de amostra do CloudWatch Logs Insights

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Insights.
3. O editor de consultas próximo do topo da tela contém uma consulta padrão que retorna os vinte eventos de log mais recentes. Acima do editor de consultas, selecione um grupo de logs para consulta.

   Quando você seleciona um grupo de CloudWatch registros, o Logs Insights detecta automaticamente os campos nos dados do grupo de registros e os exibe nos campos descobertos no painel direito. Ele também exibe um gráfico de barras de eventos de log neste grupo de logs com o passar do tempo. Esse gráfico de barras mostra a distribuição de eventos no grupo de logs correspondente à consulta e ao intervalo de tempo, e não apenas os eventos exibidos na tabela.
4. Selecione Executar consulta.

Os resultados da consulta são exibidos. Neste exemplo, os resultados são 20 eventos de log mais recentes de qualquer tipo.

5. Para ver todos os campos de um dos eventos de log retornados, selecione a seta para a esquerda desse evento de log.

Para obter mais informações sobre como executar e modificar consultas do CloudWatch Logs Insights, consulte Executar e modificar uma consulta de amostra.

Consultas de exemplo

Esta seção contém exemplos de consultas do CloudWatch Logs Insights para analisar os registros do aplicativo Managed Service for Apache Flink. Essas consultas fazem uma pesquisa entre vários exemplos de condições de erro e servem como modelos para escrever consultas que encontrem outras condições de erro.

> **Note**
>
> Substitua a região (*us-west-2*), a ID da conta (*012345678901*) e o nome do aplicativo (*YourApplication*) nos exemplos de consulta a seguir pela região do seu aplicativo e pela ID da conta.

Este tópico contém as seguintes seções:

- Analise as operações: distribuição de tarefas
- Analise as operações: mudança no paralelismo
- Analise os erros: acesso negado
- Analise os erros: fonte ou coletor não encontrado
- Analise os erros: falhas relacionadas à tarefa do aplicativo

Analise as operações: distribuição de tarefas

A consulta a seguir do CloudWatch Logs Insights retorna o número de tarefas que o Apache Flink Job Manager distribui entre os gerenciadores de tarefas. Você precisa definir o período de tempo da consulta para corresponder a uma execução de trabalho para que a consulta não retorne tarefas de trabalhos anteriores. Para obter mais informações sobre paralelismo, consulte Escalabilidade.

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

A consulta do CloudWatch Logs Insights a seguir retorna as subtarefas atribuídas a cada Gerenciador de tarefas. O número total de subtarefas é a soma do paralelismo de cada tarefa. O paralelismo de tarefas é derivado do paralelismo do operador e é igual ao paralelismo do aplicativo por padrão, a menos que você o altere no código especificando `setParallelism`. Para obter mais informações sobre como definir o paralelismo do operador, consulte Definindo o paralelismo: nível do operador na documentação do 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 obter mais informações sobre a programação de tarefas, consulte Trabalhos e programações na documentação do Apache Flink.

Analise as operações: mudança no paralelismo

A consulta a seguir do CloudWatch Logs Insights retorna alterações no paralelismo de um aplicativo (por exemplo, devido ao escalonamento automático). Essa consulta também retorna alterações manuais no paralelismo do aplicativo. Para obter mais informações sobre a escalabilidade automática, consulte the section called “Escalabilidade automática”.

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Analise os erros: acesso negado

A consulta a seguir do CloudWatch Logs Insights retorna `Access Denied` registros.

```
fields @timestamp, @message, @messageType
```

```
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\/YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Analise os erros: fonte ou coletor não encontrado

A consulta a seguir do CloudWatch Logs Insights retorna `ResourceNotFound` registros. `ResourceNotFound`os registros resultam se uma fonte ou coletor do Kinesis não for encontrada.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\/YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

Analise os erros: falhas relacionadas à tarefa do aplicativo

A consulta a seguir do CloudWatch Logs Insights retorna os registros de falhas relacionados à tarefa de um aplicativo. Esses logs acontecem se o status de um aplicativo mudar de `RUNNING` para `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
```

Para aplicativos que usam a versão 1.8.2 ou anterior do Apache Flink, as falhas relacionadas às tarefas resultarão na mudança do status do aplicativo de `RUNNING` para `FAILED`. Ao usar o Apache Flink 1.8.2 ou anterior, use a consulta a seguir para pesquisar falhas relacionadas à tarefas do aplicativo:

```
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
```

Visualizando métricas e dimensões no Managed Service for Apache Flink

Este tópico contém as seguintes seções:

Quando seu serviço gerenciado para Apache Flink processa uma fonte de dados, o serviço gerenciado para Apache Flink reporta as seguintes métricas e dimensões para a Amazon. CloudWatch

Métricas de aplicativo

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`backPressuredTimeMsPerSecond*`	Milissegundos	O tempo (em milissegundos) em que essa tarefa ou operador é contrapressionado por segundo.	Tarefa, operador, paralelismo	*Disponível para aplicativos Managed Service para Apache Flink executando somente a versão 1.13 do Flink. Essas métricas podem ser	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				úteis para identificar gargalos em um aplicativo.	
busyTimeMsPerSecond*	Milissegundos	O tempo (em milissegundos) em que essa tarefa ou operador está ocupado (sem inatividade ou contrapressão) por segundo. Pode ser NaN, se o valor não puder ser calculado.	Tarefa, operador, paralelismo	*Disponível para aplicativos Managed Service para Apache Flink executando somente a versão 1.13 do Flink. Essas métricas podem ser úteis para identificar gargalos em um aplicativo.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`cpuUtilization`	Porcentagem	Porcentagem geral de utilização da CPU nos gerenciadores de tarefas. Por exemplo, se houver cinco gerenciadores de tarefas, o Managed Service for Apache Flink publica cinco amostras dessa métrica por intervalo de geração de relatórios.	Aplicativo	Você pode usar essa métrica para monitorar a utilização mínima, média e máxima da CPU em seu aplicativo. A `CPUUtilization` métrica considera apenas o uso da CPU do processo TaskManager JVM executado dentro do contêiner.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`containerCPUUtilization`	Porcentagem	Porcentagem geral de utilização da CPU em contêineres do gerenciador de tarefas no cluster de aplicativos do Flink. Por exemplo, se houver cinco gerenciadores de tarefas, correspondentemente, há cinco TaskManager contêineres e o Managed Service for Apache Flink publica 2 x cinco amostras dessa métrica por intervalo de relatório de 1 minuto.	Aplicativo	É calculado por contêiner como: Tempo total da CPU (em segundos) consumido pelo contêiner × 100 / Limite da CPU para o contêiner (em CPUs/segundos) A `CPUUtilization` métrica considera apenas o uso da CPU do processo TaskManager JVM executado dentro do contêiner. Há outros componentes em execução fora da JVM dentro do mesmo contêiner. A métrica	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				`containerCPUUtilization` fornece uma visão mais completa, incluindo todos os processos, em termos de esgotamento da CPU no contêiner e falhas resultantes disso.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`containerMemoryUtilization`	Porcentagem	Porcentagem geral de utilização da memória nos contêineres do gerenciador de tarefas no cluster de aplicativos do Flink. Por exemplo, se houver cinco gerenciadores de tarefas, correspondentemente, há cinco TaskManager contêineres e o Managed Service for Apache Flink publica 2 x cinco amostras dessa métrica por intervalo de relatório de 1 minuto.	Aplicativo	É calculado por contêiner como: Uso de memória do contêiner (bytes) × 100 / Limite de memória do contêiner de acordo com a especificação de implantação do pod (em bytes) As `ManagedMemoryUtilzations` métricas `HeapMemoryUtilization` e consideram apenas métricas de memória específicas, como uso de memória em pilha da TaskManager JVM ou

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				memória gerenciada (uso de memória fora da JVM para processos nativos, como o RocksDB State Backend). A métrica `containerMemoryUtilization` fornece uma visão mais completa ao incluir a memória do conjunto de trabalho, que é um rastreador melhor do esgotamento total da memória. Após sua exaustão, isso resultará na `Out of Memory Error` cápsula. TaskManager	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`containerDiskUtilization`	Porcentagem	Porcentagem geral de utilização do disco nos contêineres do gerenciador de tarefas no cluster de aplicativos do Flink. Por exemplo, se houver cinco gerenciadores de tarefas, correspondentemente, há cinco TaskManager contêineres e o Managed Service for Apache Flink publica 2 x cinco amostras dessa métrica por intervalo de relatório de 1 minuto.	Aplicativo	É calculado por contêiner como: Uso do disco em bytes × 100 / Limite do disco por contêiner em bytes Para contêineres, ele representa a utilização do sistema de arquivos no qual o volume raiz do contêiner está configurado.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`currentInputWatermark`	Milissegundos	A última marca d'água que este aplicativo/operador/tarefa/thread recebeu	Aplicativo, operador, tarefa, paralelismo	Esse registro só é emitido para dimensões com duas entradas. Esse é o valor mínimo das últimas marcas d'água recebidas.	
`currentOutputWatermark`	Milissegundos	A última marca d'água que este aplicativo/operador/tarefa/thread emitiu	Aplicativo, operador, tarefa, paralelismo		

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`downtime`	Milissegundos	Para trabalhos que, atualmente, estão em situação de falha/recuperação, o tempo acabou durante essa interrupção.	Aplicativo	Essa métrica mede o tempo transcorrido enquanto um trabalho está falhando ou se recuperando. Essa métrica retorna 0 para trabalhos em execução e –1 para trabalhos concluídos. Se essa métrica não for 0 ou –1, isso indica que a tarefa do Apache Flink para o aplicativo falhou na execução.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`fullRestarts`	Contagem	O número total de vezes em que esse trabalho foi totalmente reiniciado desde que foi enviado. Essa métrica não mede reinicializações refinadas.	Aplicativo	Você pode usar essa métrica para avaliar a integrida de geral do aplicativo. As reinicial izações podem ocorrer durante a manutençã o interna do Managed Service for Apache Flink. Reinicial izações acima do normal podem indicar um problema com o aplicativ o.

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
heapMemoryUtilization	Porcentagem	Utilização geral da memória heap em todos os gerenciadores de tarefas. Por exemplo, se houver cinco gerenciadores de tarefas, o Managed Service for Apache Flink publica cinco amostras dessa métrica por intervalo de geração de relatórios.	Aplicativo	Você pode usar essa métrica para monitorar a utilização mínima, média e máxima da memória heap em seu aplicativo. A heapMemoryUtilization única conta para métricas de memória específicas, como o uso de memória em pilha da TaskManager JVM.

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`idleTimeMsPerSecond*`	Milissegundos	O tempo (em milissegundos) em que essa tarefa ou operador fica inativo (não tem dados para processar) por segundo. O tempo sem atividade exclui o tempo de contrapressão, portanto, se a tarefa for contrapressionada, ela não estará sem atividade.	Tarefa, operador, paralelismo	*Disponível para aplicativos Managed Service para Apache Flink executando somente a versão 1.13 do Flink. Essas métricas podem ser úteis para identificar gargalos em um aplicativo.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`lastCheckpointSize`	Bytes	O tamanho total do último ponto de verificação	Aplicativo	Você pode usar essa métrica para determinar a utilização do armazenamento de aplicativos em execução. Se o valor dessa métrica estiver aumentando, isso pode indicar que há um problema com seu aplicativo, como um vazamento de memória ou gargalo.

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`lastCheckpointDuration`	Milissegundos	O tempo necessário para concluir o último ponto de verificação	Aplicativo	Essa métrica mede o tempo necessário para concluir o ponto de verificação mais recente. Se o valor dessa métrica estiver aumentando, isso pode indicar que há um problema com seu aplicativo, como um vazamento de memória ou gargalo. Em alguns casos, você pode solucionar esse problema desativando o ponto de verificação.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
managedMemoryUsed*	Bytes	A quantidade de memória gerenciada usada no momento.	Aplicativo, operador, tarefa, paralelismo	*Disponível para aplicativos Managed Service para Apache Flink executando somente a versão 1.13 do Flink. Isso está relacionado à memória gerenciada pelo Flink fora da heap do Java. Ele é usado para o RocksDB State Backend e também está disponível para aplicativos.

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`managedMemoryTotal` *	Bytes	A quantidade total de memória gerenciada.	Aplicativo, operador, tarefa, paralelismo	*Disponível para aplicativos Managed Service para Apache Flink executando somente a versão 1.13 do Flink. Isso está relacionado à memória gerenciada pelo Flink fora da heap do Java. Ele é usado para o RocksDB State Backend e também está disponível para aplicativos. A métrica `ManagedMemoryUtilzations` só considera métricas de memória específicas, como a memória gerenciad

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
				a (uso de memória fora da JVM para processos nativos, como o RocksDB State Backend)
`managedMemoryUtilization*`	Porcentagem	Derivado por managedMemoryUsed/ managedMemoryTotal	Aplicativo, operador, tarefa, paralelismo	*Disponível para aplicativos Managed Service para Apache Flink executando somente a versão 1.13 do Flink. Isso está relacionado à memória gerenciada pelo Flink fora da heap do Java. Ele é usado para o RocksDB State Backend e também está disponível para aplicativos.

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`numberOfFailedCheckpoints`	Contagem	O número de vezes que o ponto de verificação falhou.	Aplicativo	Você pode usar essa métrica para monitorar a integridade e o progresso do aplicativo. Os pontos de verificação podem falhar devido a problemas do aplicativo, como problemas de throughput ou permissões.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`numRecordsIn*`	Contagem	O número total de registros que esse aplicativo, operador ou tarefa recebeu.	Aplicativo, operador, tarefa, paralelismo	*Para aplicar a estatística SUM por um período de tempo (segundo/minuto): • Selecione a métrica no nível correto. Se você estiver monitorando a métrica de um operador, precisará selecionar as métricas correspondentes do operador. • Como o Managed Service para Apache Flink faz quatro snapshots por minuto, a seguinte matemática a métrica deve ser	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				usada: m1/4, onde m1 é a estatística SUM durante um período (segundo/ minuto) O nível da métrica especific a se essa métrica mede o número total de registros que o aplicativ o todo, um operador específico ou uma tarefa específica recebeu.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`numRecordsInPerSecond*`	Contagem/segundo	O número total de registros que esse aplicativo, operador ou tarefa recebeu por segundo.	Aplicativo, operador, tarefa, paralelismo	*Para aplicar a estatística SUM por um período de tempo (segundo/minuto): • Selecione a métrica no nível correto. Se você estiver monitorando a métrica de um operador, precisará selecionar as métricas correspondentes do operador. • Como o Managed Service para Apache Flink faz quatro snapshots por minuto, a seguinte matemática a métrica deve ser	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				usada: m1/4, onde m1 é a estatística SUM durante um período (segundo/ minuto) O nível da métrica especific a se essa métrica mede o número total de registros que o aplicativ o todo, um operador específico ou uma tarefa específica recebeu por segundo.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`numRecordsOut*`	Contagem	O número total de registros que esse aplicativo, operador ou tarefa emitiu.	Aplicativo, operador, tarefa, paralelismo	*Para aplicar a estatística SUM por um período de tempo (segundo/minuto): • Selecione a métrica no nível correto. Se você estiver monitorando a métrica de um operador, precisará selecionar as métricas correspondentes do operador. • Como o Managed Service para Apache Flink faz quatro snapshots por minuto, a seguinte matemática a métrica deve ser	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				usada: m1/4, onde m1 é a estatística SUM durante um período (segundo/ minuto) O nível da métrica especific a se essa métrica mede o número total de registros que o aplicativ o todo, um operador específico ou uma tarefa específica emitiu.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`numLateRecordsDropped*`	Contagem	Aplicativo, operador, tarefa, paralelismo		*Para aplicar a estatística SUM por um período de tempo (segundo/minuto): • Selecione a métrica no nível correto. Se você estiver monitorando a métrica de um operador, precisará selecionar as métricas correspondentes do operador. • Como o Managed Service para Apache Flink faz quatro snapshots por minuto, a seguinte matemática a métrica deve ser	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				usada: m1/4, onde m1 é a estatística SUM durante um período (segundo/ minuto) O número de registros que esse operador ou tarefa reduziu devido ao atraso na chegada.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`numRecordsOutPerSecond*`	Contagem/segundo	O número total de registros que esse aplicativo, operador ou tarefa emitiu por segundo.	Aplicativo, operador, tarefa, paralelismo	*Para aplicar a estatística SUM por um período de tempo (segundo/minuto): • Selecione a métrica no nível correto. Se você estiver monitorando a métrica de um operador, precisará selecionar as métricas correspondentes do operador. • Como o Managed Service para Apache Flink faz quatro snapshots por minuto, a seguinte matemática a métrica deve ser

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				usada: m1/4, onde m1 é a estatística SUM durante um período (segundo/minuto) O nível da métrica especific a se essa métrica mede o número total de registros que o aplicativo todo, um operador específico ou uma tarefa específica emitiu por segundo.	
`oldGenerationGCCount`	Contagem	O número total de operações antigas de coleta de resíduos que ocorreram em todos os gerenciadores de tarefas.	Aplicativo		

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`oldGenerationGCTime`	Milissegundos	O tempo total gasto executando operações antigas de coleta de resíduos.	Aplicativo	Você pode usar essa métrica para monitorar a soma, a média e o tempo máximo de coleta de resíduos.
`threadCount`	Contagem	O número total de threads ativos usados pelo aplicativo.	Aplicativo	Essa métrica mede o número de segmentos usados pelo código do aplicativo. Isso não é o mesmo que paralelismo de aplicativos.
`uptime`	Milissegundos	O tempo no qual o trabalho foi executado sem interrupções.	Aplicativo	Você pode usar essa métrica para determinar se um trabalho está sendo executado com êxito. Essa métrica retorna –1 para trabalhos concluídos.

Métricas do conector do Kinesis Data Streams

AWS emite todos os registros do Kinesis Data Streams, além dos seguintes:

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`millisbehindLatest`	Milissegundos	O número de milissegundos em que o consumidor está atrás do início do fluxo de dados, indicando o quão atrasado o consumidor está em relação ao horário atual.	Aplicação (para Stream), Paralelismo (para) ShardId	• Um valor zero indica que o processamento de registros foi alcançado e não há nenhum registro novo para processar no momento. A métrica de um fragmento específico pode ser especificada pelo nome do fluxo e pelo ID do fragmento. • Um valor de –1 indica que o serviço ainda não relatou um valor para a métrica.
`bytesRequestedPerFetch`	Bytes	Os bytes solicitados em uma única	Aplicação (para Stream), Paralelismo (para) ShardId	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
		chamada para `getRecords` .		

Métrica do Amazon MSK Connector

AWS emite todos os registros do Amazon MSK, além dos seguintes:

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`currentoffsets`	N/D	O deslocamento de leitura atual do consumidor, para cada partição. A métrica de uma partição específica pode ser especificada pelo nome do tópico e pela ID da partição.	Aplicação (para tópico), paralelismo (para) PartitionId	
`commitsFailed`	N/D	O número total de falhas de confirmação de deslocamentos para o Kafka, se o deslocamento e o ponto de verificação estiverem habilitados.	Aplicativo, operador, tarefa, paralelismo	Enviar os deslocamentos de volta ao Kafka é apenas um meio de expor o progresso do consumidor, portanto, uma falha de confirmação

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
				não afeta a integridade dos deslocamentos de partição do ponto de verificação do Flink.
`commitsSucceeded`	N/D	O número total de confirmações de deslocamentos bem-sucedidas para o Kafka, se a confirmação do deslocamento e o ponto de verificação estiverem habilitados.	Aplicativo, operador, tarefa, paralelismo	
`committedoffsets`	N/D	Os últimos deslocamentos confirmados com sucesso para o Kafka, para cada partição. A métrica de uma partição específica pode ser especificada pelo nome do tópico e pela ID da partição.	Aplicação (para tópico), paralelismo (para) PartitionId	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`records_lag_max`	Contagem	O atraso máximo em termos de número de registros para qualquer partição nesta janela	Aplicativo, operador, tarefa, paralelismo	
`bytes_consumed_rate`	Bytes	O número médio de bytes consumidos por segundo para um tópico	Aplicativo, operador, tarefa, paralelismo	

Métricas do Apache Zeppelin

Para notebooks que usam o Studio, AWS emite as seguintes métricas no nível do aplicativo: `KPUs`, `cpuUtilization`, `heapMemoryUtilization`, `oldGenerationGCTime`, `oldGenerationGCCount` e `threadCount`. Além disso, ela emite as métricas mostradas na tabela a seguir, também no nível do aplicativo.

Métrica	Unidade	Descrição	Nome no Prometheus
`zeppelinCpuUtilization`	Porcentagem	Porcentagem geral de utilização da CPU no servidor Apache Zeppelin.	`process_cpu_usage`
`zeppelinHeapMemoryUtilization`	Porcentagem	Porcentagem geral de utilização da memória heap para o servidor Apache Zeppelin.	`jvm_memory_used_bytes`

Métrica	Unidade	Descrição	Nome no Prometheus
`zeppelinThreadCount`	Contagem	O número total de threads ativos usados pelo servidor Apache Zeppelin.	`jvm_threads_live_threads`
`zeppelinWaitingJobs`	Contagem	O número de trabalhos enfileirados do Apache Zeppelin esperando por um thread.	`jetty_threads_jobs`
`zeppelinServerUptime`	Segundos	O tempo total em que o servidor esteve ativo e funcionando.	`process_uptime_seconds`

Visualizando CloudWatch métricas

Você pode visualizar CloudWatch as métricas do seu aplicativo usando o CloudWatch console da Amazon ou AWS CLI o.

Para visualizar métricas usando o CloudWatch console

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Métricas.
3. No painel CloudWatch Métricas por categoria do Managed Service for Apache Flink, escolha uma categoria de métricas.
4. No painel superior, role para visualizar a lista completa de métricas.

Para visualizar métricas usando a AWS CLI

- Em um prompt de comando, use o seguinte comando.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Definindo níveis de relatórios de CloudWatch métricas

Você pode controlar o nível das métricas do aplicativo que seu aplicativo cria. O Managed Service for Apache Flink oferece suporte para os seguintes níveis de métricas:

- Aplicativo: o aplicativo relata apenas o nível mais alto de métricas para cada aplicativo. Por padrão, as métricas do Managed Service for Apache Flink são publicadas no nível do aplicativo.
- Tarefa: o aplicativo relata dimensões métricas específicas da tarefa para métricas definidas com o nível de relatório métrico da tarefa, como o número de registros de entrada e saída do aplicativo por segundo.
- Operador: o aplicativo relata dimensões métricas específicas do operador para métricas definidas com o nível de relatório métrico do operador, como métricas para cada operação de filtro ou mapa.
- Paralelismo: o aplicativo relata `Task` e `Operator` métricas de nível para cada thread de execução. O nível de relatório não é recomendado para aplicativos com uma configuração de paralelismo acima de 64 devido aos custos excessivos.

> **Note**
>
> Você só deve usar esse nível métrico para solucionar problemas devido à quantidade de dados métricos que o serviço gera. Você só pode definir esse nível métrico usando a CLI. Esse nível métrico não está disponível no console.

O nível padrão é Aplicativo. O aplicativo relata métricas no nível atual e em todos os níveis superiores. Por exemplo, se o nível de relatório estiver definido como Operator, o aplicativo reportará as métricas de Aplicativo, Tarefa e Operador.

Você define o nível do relatório de CloudWatch métricas usando o `MonitoringConfiguration` parâmetro da `CreateApplication`ação ou o `MonitoringConfigurationUpdate` parâmetro da `UpdateApplication`ação. O exemplo a seguir de solicitação para a `UpdateApplication`ação define o nível de relatório de CloudWatch métricas como Tarefa:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
```

```
            "ConfigurationTypeUpdate": "CUSTOM",
            "MetricsLevelUpdate": "TASK"
         }
      }
   }
}
```

Você também pode configurar o nível de registro em log usando o parâmetro `LogLevel` da ação `CreateApplication` ou o parâmetro `LogLevelUpdate` da ação `UpdateApplication`. Você pode usar os seguintes níveis de log:

- `ERROR`: registra eventos de erro potencialmente recuperáveis.
- `WARN`: registra eventos de alerta que podem levar a um erro.
- `INFO`: registra eventos informativos.
- `DEBUG`: registra eventos gerais de depuração.

Para obter mais informações sobre os níveis de registro em log do Log4j, consulte Níveis de registro personalizados na documentação do Apache Log4j.

Usando métricas personalizadas com o Amazon Managed Service for Apache Flink

O Managed Service for Apache Flink expõe 19 métricas CloudWatch, incluindo métricas de uso de recursos e taxa de transferência. Além disso, você pode criar suas próprias métricas para rastrear dados específicos do aplicativo, como eventos de processamento ou acesso a recursos externos.

Este tópico contém as seguintes seções:

- Como funciona
- Exemplos
- Visualização de métricas personalizadas

Como funciona

As métricas personalizadas no Managed Service for Apache Flink usam o sistema métrico Apache Flink. As métricas do Apache Flink têm os atributos a seguir:

- Tipo: o tipo de uma métrica descreve como ela mede e relata dados. Os tipos de métricas disponíveis no Apache Flink incluem Contagem, Indicador, Histograma e Medidor. Para obter mais informações sobre os tipos de métricas do Apache Flink, consulte Tipos de métricas.

  > **Note**
  >
  > AWS CloudWatch As métricas não são compatíveis com o tipo de métrica Histogram Apache Flink. CloudWatch só pode exibir métricas do Apache Flink dos tipos Count, Gauge e Meter.

- Escopo: o escopo de uma métrica consiste em seu identificador e um conjunto de pares de valores-chave que indicam como a métrica será reportada. CloudWatch O identificador de uma métrica consiste no seguinte:
  - Um escopo do sistema, que indica o nível no qual a métrica é relatada (por exemplo, Operator).
  - Um escopo de usuário, que define atributos como variáveis de usuário ou nomes de grupos de métricas. Esses atributos são definidos usando `MetricGroup.addGroup(key, value)` ou `MetricGroup.addGroup(name)`.

  Para obter mais informações sobre as métricas de escopos, consulte Escopo.

Para obter mais informações sobre a métrica do Apache Flink, consulte Métrica na documentação do Apache Flink.

Para criar uma métrica personalizada em seu Managed Service for Apache Flink, você pode acessar o sistema métrico do Apache Flink a partir de qualquer função do usuário que se estenda `RichFunction` por meio de chamadas para `GetMetricGroup`. Esse método retorna um MetricGroupobjeto que você pode usar para criar e registrar métricas personalizadas. O Managed Service for Apache Flink relata todas as métricas criadas com a chave de grupo para`KinesisAnalytics`. CloudWatch As métricas personalizadas que você define têm as seguintes características:

- Sua métrica personalizada tem um nome de métrica e um nome de grupo. Esses nomes devem conter caracteres alfanuméricos.
- Os atributos que você define no escopo do usuário (exceto para o grupo de `KinesisAnalytics` métricas) são publicados como CloudWatch dimensões.
- Por padrão, as métricas personalizadas são publicadas no nível `Application`.

- As dimensões (Task/Operator/Parallelism) são adicionadas à métrica com base no nível de monitoramento do aplicativo. Você define o nível de monitoramento do aplicativo usando o MonitoringConfigurationparâmetro da CreateApplicationação ou o MonitoringConfigurationUpdateparâmetro ou da UpdateApplicationação.

Exemplos

Os exemplos de código a seguir demonstram como criar uma classe de mapeamento que cria e incrementa uma métrica personalizada e como implementar a classe de mapeamento em seu aplicativo adicionando-a a um objeto `DataStream`.

Métrica personalizada de contagem de registros

O exemplo de código a seguir demonstra como criar uma classe de mapeamento que cria uma métrica que conta registros em um fluxo de dados (a mesma funcionalidade da 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;
        }
    }
```

No exemplo anterior, a variável `valueToExpose` é incrementada para cada registro que o aplicativo processa.

Depois de definir sua classe de mapeamento, você cria um fluxo no aplicativo que implementa o mapa:

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

Para obter o código completo desse aplicativo, consulte Aplicativo de métrica personalizada para contagem de registros.

Métrica personalizada de contagem de palavras

O exemplo de código a seguir demonstra como criar uma classe de mapeamento que cria uma métrica que conta palavras em um fluxo de dados:

```
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<Tuple2<String, Integer>>out) {
                // normalize and split the line
                String[] tokens = value.toLowerCase().split("\\W+");

                // emit the pairs
                for (String token : tokens) {
                    if (token.length() > 0) {
                        counter.inc();
                        out.collect(new Tuple2<>(token, 1));
                    }
                }
```

```
            }
        }
```

No exemplo anterior, a variável `counter` é incrementada para cada palavra que o aplicativo processa.

Depois de definir sua classe de mapeamento, você cria um fluxo no aplicativo que implementa o 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 obter o código completo desse aplicativo, consulte Aplicativo de métrica personalizada para contagem de palavras.

Visualização de métricas personalizadas

As métricas personalizadas do seu aplicativo aparecem no console de CloudWatch métricas no AWS/KinesisAnalyticspainel, no grupo de métricas do aplicativo.

Usando CloudWatch alarmes com o Amazon Managed Service para Apache Flink

Usando os alarmes CloudWatch métricos da Amazon, você assiste a uma CloudWatch métrica durante um período de tempo especificado por você. O alarme executa uma ou mais ações com base no valor da métrica ou na expressão em relação a um limite em alguns períodos. Um exemplo de uma ação é o envio de uma notificação para um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter mais informações sobre CloudWatch alarmes, consulte Usando CloudWatch alarmes da Amazon.

Alarmes recomendados

Esta seção contém os alarmes recomendados para monitorar o aplicativos Managed Service for Apache Flink.

A tabela descreve os alarmes recomendados e tem as seguintes colunas:

- Expressão métrica: a métrica ou expressão métrica a ser testada em relação ao limite.
- Estatística: a estatística usada para verificar a métrica — por exemplo, Average (Média).
- Limite: o uso deste alarme exige que você determine o limite do desempenho esperado do aplicativo. Você precisa determinar esse limite monitorando seu aplicativo em condições normais.
- Descrição: causas que podem acionar esse alarme e possíveis soluções para a situação.

Expressão da métrica	Estatística	Limite	Descrição
`Tempo de inatividade` > 0	Average	0	A downtime greater than zero indicates that the applicati on has failed. If the value is larger than 0, the application is not processing any data. Recommend ed for all applicati ons. The `Tempo de inatividade` metric measures the duration of an outage. A downtime greater than zero indicates that the application has failed. For troubleshooting, see O aplicativo está sendo reiniciado.
`TARIFA (numberOf Failedpontos de verificação)` > 0	Average	0	This metric counts the number of failed checkpoints since the application started. Depending on the

Expressão da métrica	Estatística	Limite	Descrição
			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 RATE(numberOfFailedCheckpoints) 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 processing the input data. For troubleshooting, see O ponto de verificação está atingindo o tempo limite.

Expressão da métrica	Estatística	Limite	Descrição
`Operador. numRecord sOutPerSecond` < threshold	Average	The minimum number of records emitted from the applicati on during normal conditions.	Recommended for all applications. Falling below this threshold can indicate that the applicati on isn't making expected progress on the input data. For troubleshooting, see O throughput é muito lento.

Expressão da métrica	Estatística	Limite	Descrição
`records_lag_max|millisbehindLatest` > threshold	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 `records_lag_max` metric for a Kafka source, or the `millisbehindLatest` 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

Expressão da métrica	Estatística	Limite	Descrição
			O throughput é muito lento.

Expressão da métrica	Estatística	Limite	Descrição
`lastCheckpointDuration` > threshold	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 considering monitoring the change rate with `TAXA(lastCheckpointSize)` and `TAXA(lastCheckpointDuration)`. If the `lastCheckpointDuration` continuously increases, rising above this threshold can indicate that the application isn't making expected progress on the input

Expressão da métrica	Estatística	Limite	Descrição
			data, or that there are problems with application health such as backpress ure. For troublesh ooting, see Crescimen to ilimitado do estado.

Expressão da métrica	Estatística	Limite	Descrição
`lastCheckpointSize` > threshold	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 considering monitoring the change rate with `TAXA (lastCheckpointSize)` and `TAXA (lastCheckpointDuration)` . If the `lastCheckpointSize` continuously increases, rising above this threshold can indicate that the application is accumulating state data. If the state data

Expressão da métrica	Estatística	Limite	Descrição
			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 troublesh ooting, see Crescimen to ilimitado do estado.
`heapMemor yUtilization` > 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 `heapMemor yUtilization` size during normal conditions, with a recommended value of 90 percent.	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 increasin g the application parallelism. For more information about increasing resources, see Escalabilidade.

Expressão da métrica	Estatística	Limite	Descrição
`cpuUtilization` > 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 `cpuUtilization` 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 Escalabilidade.
`threadsCount` > threshold	Maximum	The maximum expected `threadsCount` 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.

Expressão da métrica	Estatística	Limite	Descrição
`(oldGarbageCollectionTempo * 100) /60_000 durante um 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.
`TAXA (oldGarbageCollectionContagem)` > threshold	Maximum	The maximum expected `oldGarbageCollectionContagem` 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.
`Operador. currentOutputWatermark - Operador. currentInputWatermark` > 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.

Gravando mensagens personalizadas em CloudWatch registros

Você pode gravar mensagens personalizadas no log do aplicativo Managed Service for Apache Flink. CloudWatch Você faz isso usando abiblioteca log4j do Apache ou abiblioteca Simple Logging Facade for Java (SLF4J).

Tópicos

Gravar em CloudWatch registros usando o Log4J

1. Adicione as seguintes dependências ao arquivo pom.xml do aplicativo:

```
<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. Inclua o objeto da biblioteca:

```
import org.apache.logging.log4j.Logger;
```

3. Instancie o objeto Logger, transmitindo sua classe de aplicativo:

```
private static final Logger log =
 LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. Grave no log usando log.info. Um grande número de mensagens é gravado no log do aplicativo. Para facilitar a filtragem de suas mensagens personalizadas, use o nível de log INFO do aplicativo.

```
log.info("This message will be written to the application's CloudWatch log");
```

O aplicativo grava um registro no log com uma mensagem semelhante à seguinte:

```
{
  "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"
}
```

Gravar em CloudWatch registros usando SLF4J

1. Adicione as seguintes dependências ao arquivo `pom.xml` do aplicativo:

   ```
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.7</version>
       <scope>runtime</scope>
   </dependency>
   ```

2. Inclua os objetos da biblioteca:

   ```
   import org.slf4j.Logger;
   import org.slf4j.LoggerFactory;
   ```

3. Instancie o objeto `Logger`, transmitindo sua classe de aplicativo:

   ```
   private static final Logger log =
    LoggerFactory.getLogger(YourApplicationClass.class);
   ```

4. Grave no log usando `log.info`. Um grande número de mensagens é gravado no log do aplicativo. Para facilitar a filtragem de suas mensagens personalizadas, use o nível de log `INFO` do aplicativo.

   ```
   log.info("This message will be written to the application's CloudWatch log");
   ```

O aplicativo grava um registro no log com uma mensagem semelhante à seguinte:

```
{
  "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"
}
```

Registrando em log as chamadas de API para o Managed Service for Apache Flink com AWS CloudTrail

O Managed Service for Apache Flink é integrado comAWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Managed Service for Apache Flink. CloudTrail captura todas as chamadas de API para o Managed Service for Apache Flink como eventos. As chamadas capturadas incluem as chamadas do console do Managed Service for Apache Flink e as chamadas de código para as operações de API do Managed Service for Apache Flink. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o Managed Service for Apache Flink. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Managed Service for Apache Flink, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Serviço gerenciado para informações do Apache Flink em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Managed Service for Apache Flink, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. É possível visualizar, pesquisar e baixar os eventos recentes em sua conta da AWS. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para obter um registro de eventos em andamento na sua conta da AWS, incluindo eventos do Managed Service for Apache Flink, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra em log eventos de todas as Regiões na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- Visão Geral para Criar uma Trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e recebendo arquivos de CloudTrail log de várias contas

Todas as ações do Managed Service for Apache Flink são registradas CloudTrail e documentadas na referência da API Managed Service for Apache Flink. Por exemplo, chamadas para as `UpdateApplication` ações `CreateApplication` e geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou usuário do IAM AWS Identity and Access Management
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

Entendendo as entradas no arquivo de log do Managed Service for Apache Flink

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações

sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra as DescribeApplicationações AddApplicationCloudWatchLoggingOptione.

```
{
    "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": "kinesisanlaytics.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": "kinesisanlaytics.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"
        }
    ]
}
```

Ajustar o desempenho do Amazon Managed Service for Apache Flink

Este tópico descreve técnicas para monitorar e melhorar o desempenho do seu aplicativo Managed Service for Apache Flink.

Tópicos

Solução de problemas de desempenho

Esta seção contém uma lista de sintomas que você pode verificar para diagnosticar e corrigir problemas de desempenho.

Se sua fonte de dados for um stream do Kinesis, os problemas de desempenho geralmente se apresentam como uma métrica alta ou crescente`millisbehindLatest`. Para outras fontes, você pode verificar uma métrica semelhante que representa o atraso na leitura da fonte.

O caminho dos dados

Ao investigar um problema de desempenho com seu aplicativo, considere todo o caminho que seus dados percorrem. Os seguintes componentes do aplicativo podem se tornar gargalos de desempenho e criar contrapressão se não forem projetados ou provisionados adequadamente:

- Fontes de dados e destinos: garanta que os recursos externos com os quais seu aplicativo interage sejam propriedade provisionada para o throughput que seu aplicativo experimentará.
- Dados do estado: certifique-se de que seu aplicativo não interaja com o armazenamento do estado com muita frequência.

  Você pode otimizar o serializador que seu aplicativo está usando. O serializador Kryo padrão pode lidar com qualquer tipo serializável, mas você pode usar um serializador com melhor desempenho se seu aplicativo armazenar dados apenas em tipos POJO. Para obter informações sobre serializadores Apache Flink, consulte Tipos de dados e serialização na documentação do Apache Flink.

- Operadores: garanta que a lógica de negócios implementada por seus operadores não seja muito complicada ou que você não esteja criando ou usando recursos com cada registro processado. Além disso, certifique-se de que seu aplicativo não esteja criando janelas deslizantes ou giratórias com muita frequência.

Soluções de solução de problemas de desempenho

Esta seção contém possíveis soluções para problemas de desempenho.

Tópicos

Níveis de monitoramento do CloudWatch

Verifique se os níveis de monitoramento do CloudWatch não estão definidos como uma configuração muito detalhada.

A `Debug` configuração do nível de log de monitoramento gera uma grande quantidade de tráfego, o que pode criar contrapressão. Você só deve usá-lo enquanto estiver investigando ativamente os problemas com o aplicativo.

Se seu aplicativo tiver uma `Parallelism` configuração alta, o uso do `Parallelism` Nível de Métricas de Monitoramento também gerará uma grande quantidade de tráfego que pode causar contrapressão. Use esse nível de métrica somente quando `Parallelism` seu aplicativo estiver baixo ou ao investigar problemas com o aplicativo.

Para obter mais informações, consulte Níveis de monitoramento de aplicativos.

Métricas da CPU do aplicativo

Verifique a `CPU` métrica do aplicativo. Se essa métrica estiver acima de 75 por cento, você pode permitir que o aplicativo aloque mais recursos para si mesmo ativando o ajuste de escala automático.

Se o ajuste de escala automático estiver ativado, o aplicativo aloca mais recursos se o uso da CPU for superior a 75% por 15 minutos. Para obter mais informações sobre escalonamento, consulte a seção Gerenciar a escalabilidade de forma adequada a seguir, e Escalabilidade.

> **Note**
>
> Um aplicativo só será escalado automaticamente em resposta ao uso da CPU. O aplicativo não será escalado automaticamente em resposta a outras métricas do sistema, como`heapMemoryUtilization`. Se seu aplicativo tiver um alto nível de uso de outras métricas, aumente o paralelismo do aplicativo manualmente.

Paralelismo de aplicativos

Aumente o paralelismo do aplicativo. Você atualiza o paralelismo do aplicativo usando o `ParallelismConfigurationUpdate` parâmetro da UpdateApplication.

O máximo de KPUs para um aplicativo é 64 por padrão e pode ser aumentado solicitando um aumento de limite.

Também é importante atribuir paralelismo a cada operador com base em seu workload, em vez de aumentar apenas o paralelismo do aplicativo. Veja Paralelismo do operador a seguir.

Log de aplicativo

Verifique se o aplicativo está registrando uma entrada para cada log que está sendo processado. Gravar uma entrada de log para cada registro nos momentos em que o aplicativo tem alto throughput causará graves gargalos no processamento de dados. Para verificar essa condição, consulte seus logs em busca de entradas de logs que seu aplicativo grava com cada registro que ele processa. Para mais informações sobre a leitura de logs de aplicativo, consulte the section called “Analisando logs”.

Paralelismo do operador

Verifique se o workload do seu aplicativo está distribuído uniformemente entre os processos de trabalho.

Para obter informações sobre como ajustar o workload dos operadores do seu aplicativo, consulteEscalonamento operador.

Lógica do aplicativo

Examine a lógica do seu aplicativo em busca de operações ineficientes ou sem desempenho, como acessar uma dependência externa (como um banco de dados ou um serviço web), acessar o estado do aplicativo etc. Uma dependência externa também pode prejudicar o desempenho se não tiver desempenho ou não for acessível de forma confiável, o que pode fazer com que a dependência externa retorne erros. `HTTP 500`

Se seu aplicativo usa uma dependência externa para enriquecer ou processar dados recebidos, considere usar E/S assíncrona em vez disso. Para obter mais informações, consulte E/S assíncrona na Documentação do Apache Flink.

Memória do aplicativo

Verifique se há vazamentos de recursos em seu aplicativo. Se seu aplicativo não estiver descartando adequadamente os encadeamentos ou a memória, você poderá ver `millisbehindLatest`, `CheckpointSize` e `CheckpointDuration` o aumento ou o aumento gradual da métrica. Essa condição também pode levar a falhas no gerenciador de tarefas ou no gerenciador de tarefas.

Melhores práticas de desempenho

Esta seção descreve considerações especiais para projetar um aplicativo para desempenho.

Gerenciar a escalabilidade de forma adequada

Esta seção contém informações sobre como gerenciar a escalabilidade em nível de aplicativo e de operador.

Esta seção contém os seguintes tópicos:

- Gerenciar o escalonamento de aplicativos adequadamente
- Gerencie o escalonamento do operador adequadamente

Gerenciar o escalonamento de aplicativos adequadamente

Você pode usar o ajuste de escala automático para lidar com picos inesperados na atividade do aplicativo. Os KPUs do seu aplicativo aumentarão automaticamente caso os seguintes critérios sejam atendidos:

- O ajuste de escala automático está ativado para o aplicativo.
- O uso da CPU permanece acima de 75 por cento por 15 minutos.

Se o automáticoajuste de escala automático estiver ativado, mas o uso da CPU não permanecer nesse limite, o aplicativo não aumentará a escala verticalmente das KPUs. Se você tiver um aumento no uso da CPU que não atinge esse limite ou um aumento em uma métrica de uso diferente, como, por exemplo`heapMemoryUtilization`, aumente a escala manualmente para permitir que seu aplicativo gerencie picos de atividade.

> **Note**
>
> Se o aplicativo tiver adicionado automaticamente mais recursos por meio do ajuste de escala automático, o aplicativo liberará os novos recursos após um período de inatividade. A redução de recursos afetará temporariamente o desempenho.

Para ter mais informações sobre escalonamento, consulte Escalabilidade.

Gerencie o escalonamento do operador adequadamente

Você pode melhorar o desempenho do seu aplicativo verificando se o workload do seu aplicativo está distribuído uniformemente entre os processos de trabalho e se os operadores do seu aplicativo têm os recursos do sistema de que precisam para serem estáveis e com desempenho.

Você pode definir o paralelismo para cada operador no código do seu aplicativo usando a configuração `parallelism`. Se você não definir o paralelismo para um operador, ele usará a configuração de paralelismo no nível do aplicativo. Os operadores que usam a configuração de paralelismo no nível do aplicativo podem potencialmente usar todos os recursos do sistema disponíveis para o aplicativo, tornando-o instável.

Para determinar melhor o paralelismo de cada operador, considere os requisitos relativos de recursos do operador em comparação com os outros operadores no aplicativo. Defina operadores

que consomem mais recursos para uma configuração de paralelismo de operadores mais alta do que operadores que consomem menos recursos.

O paralelismo total do operador para o aplicativo é a soma do paralelismo para todos os operadores no aplicativo. Você ajusta o paralelismo total do operador para seu aplicativo determinando a melhor proporção entre ele e o total de slots de tarefas disponíveis para seu aplicativo. Uma proporção estável típica do paralelismo total do operador em relação aos slots de tarefas é de 4:1, ou seja, o aplicativo tem um slot de tarefa disponível para cada quatro subtarefas do operador disponíveis. Um aplicativo com operadores que consomem mais recursos pode precisar de uma proporção de 3:1 ou 2:1, enquanto um aplicativo com operadores que consomem menos recursos pode ser estável com uma proporção de 10:1.

Você pode definir a proporção para o operador usando Propriedades de runtime, para poder ajustar o paralelismo do operador sem compilar e carregar o código do aplicativo.

O exemplo de código a seguir demonstra como definir o paralelismo do operador como uma proporção ajustável do paralelismo atual do aplicativo:

```
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

 applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

Para obter informações sobre subtarefas, slots de tarefas e outros recursos do aplicativo, consulteRecursos do aplicativo.

Para controlar a distribuição do workload nos processos de trabalho do seu aplicativo, use a configuração `Parallelism` e o método de partição `KeyBy` . Para obter mais informações, consulte os seguintes tópicos na documentação do Apache Flink:

- Execução paralela
- Transformações do DataStream

Monitore o uso de recursos de dependência externa

Se houver um gargalo de desempenho em um destino (como Kinesis Streams, Kinesis Data Firehose, DynamoDB ou OpenSearch Service), seu aplicativo sofrerá contrapressão. Verifique se suas dependências externas estão adequadamente provisionadas para o throughput do seu aplicativo.

> **Note**
>
> Falhas em outros serviços podem causar falhas em seu aplicativo. Se você estiver vendo falhas em seu aplicativo, verifique se há falhas nos logs do CloudWatch dos seus serviços de destino.

Execute seu aplicativo Apache Flink localmente

Para solucionar problemas de memória, você pode executar seu aplicativo em uma instalação local do Flink. Isso lhe dará acesso a ferramentas de depuração, como rastreamento de pilha e despejos de heap, que não estão disponíveis ao executar seu aplicativo no Managed Service for Apache Flink.

Para obter informações sobre como criar uma instalação local do Flink, consulte o Tutorial de configuração local na documentação do Apache Flink.

Monitorar o desempenho

Esta seção descreve as ferramentas para monitorar o desempenho de um aplicativo.

Monitorar desempenho usando as métricas do CloudWatch

Você monitora o uso de recursos, o throughput, o ponto de verificação e o tempo de inatividade do seu aplicativo usando as métricas do CloudWatch. Para obter informações sobre como usar as métricas do CloudWatch com seu aplicativo Managed Service for Apache Flink, consulte. Métricas e dimensões no Managed Service for Apache Flink

Monitorar o desempenho usando CloudWatch Logs e alarmes

Você monitora condições de erro que poderiam causar problemas de desempenho usando o CloudWatch Logs.

As condições de erro aparecem nas entradas de registro quando o status do trabalho do Apache Flink muda de status `RUNNING` para status `FAILED`.

Você usa os alarmes do CloudWatch para criar notificações sobre problemas de desempenho, como uso de recursos ou métricas de pontos de verificação acima de um limite seguro ou alterações inesperadas no status do aplicativo.

Para obter informações sobre a criação de alarmes do CloudWatch para um aplicativo Managed Service for Apache Flink, consulte. Alarmes

Quota do Managed Service for Apache Flink e do bloco de anotações do Studio

Ao trabalhar com o Amazon Managed Service for Apache Flink, observe a quota a seguir:

- É possível criar até 50 aplicativos do Managed Service for Apache Flink por região na sua conta. Você pode criar um caso para solicitar aplicativos adicionais por meio do formulário de aumento de Service Quotas. Para obter informações, consulte o Centro da AWS Support.

  Para obter uma lista de regiões que oferecem suporte ao Managed Service for Apache Flink, consulte Regiões e endpoints do Managed Service for Apache Flink.

- A quantidade máxima de unidades de processamento do Kinesis (KPU) é limitado a 64 por padrão. Para obter instruções sobre como solicitar um aumento dessa quota, consulte Para solicitar um aumento de quota em Service Quotas. Certifique-se de especificar o prefixo do aplicativo ao qual o novo limite de KPU precisa ser aplicado.

  Com o Managed Service for Apache Flink, sua conta AWS é cobrada pelos recursos alocados, em vez dos recursos que seu aplicativo usa. É cobrada uma taxa por hora com base no número máximo de KPUs usadas para executar o aplicativo de processamento de streams. Uma única KPU oferece 1 vCPU e 4 GiB de memória. Para cada KPU, o serviço também provisiona 50 GiB de armazenamento de aplicativos em execução.

- É possível criar até 1.000 Snapshots do Managed Service for Apache Flink por aplicativo.

- Você pode atribuir até 50 tags por aplicativo.

- O tamanho máximo de um arquivo JAR do aplicativo é 512 MiB. Se você exceder essa quota, seu aplicativo não será iniciado.

Para blocos de anotações do Studio, as quotas a seguir são aplicáveis. Para solicitar uma quota maior, crie um caso de suporte.

- `websocketMessageSize` = 5 MiB
- `noteSize` = 5 MiB
- `noteCount` = 1000
- `Max cumulative UDF size` = 100 MiB
- `Max cumulative dependency jar size` = 300 MiB

Manutenção do Managed Service for Apache Flink

O Managed Service for Apache Flink corrige seus aplicativos periodicamente com atualizações de segurança do sistema operacional e da imagem do contêiner para manter a conformidade e atingir as metas de segurança AWS. A tabela a seguir lista a janela de tempo padrão durante a qual o Managed Service for Apache Flink executa esse tipo de manutenção. A manutenção do seu aplicativo pode ocorrer a qualquer momento durante a janela de tempo correspondente à sua região. Seu aplicativo pode passar por um tempo de inatividade de 10 a 30 segundos durante esse processo de manutenção. No entanto, a duração real do tempo de inatividade depende do estado do aplicativo. Para obter informações sobre como minimizar o impacto desse tempo de inatividade, consulte the section called “Tolerância a falhas: pontos de verificação e pontos de salvamento”.

Para alterar a janela de tempo durante a qual o Managed Service for Apache Flink realiza a manutenção em seu aplicativo, use a API UpdateApplicationMaintenanceConfiguration.

Região	Janela de tempo de manutenção
AWS GovCloud (Oeste dos EUA)	6h às 14h (UTC)
AWS GovCloud (Leste dos EUA)	3h às 11h (UTC)
Leste dos EUA (N. da Virgínia)	3h às 11h (UTC)
Leste dos EUA (Ohio)	3h às 11h (UTC)
Oeste dos EUA (N. da Califórnia)	6h às 14h (UTC)
Oeste dos EUA (Oregon)	6h às 14h (UTC)
Ásia-Pacífico (Hong Kong)	De 13h às 21h (UTC)
Ásia-Pacífico (Mumbai)	De 16:30 a 00:30 (UTC)
Ásia-Pacífico (Hyderabad)	De 16:30 a 00:30 (UTC)
Ásia-Pacífico (Seul)	De 13h às 21h (UTC)
Ásia-Pacífico (Singapura)	De 14:00 a 22:00 (UTC)

Região	Janela de tempo de manutenção
Ásia-Pacífico (Sydney)	12h às 20h (UTC)
Ásia-Pacífico (Jacarta)	Das 15h às 23h (UTC)
Ásia-Pacífico (Tóquio)	De 13h às 21h (UTC)
Canadá (Central)	3h às 11h (UTC)
China (Pequim)	De 13h às 21h (UTC)
China (Ningxia)	De 13h às 21h (UTC)
Europa (Frankfurt)	6h às 14h (UTC)
Europa (Zurique)	De 20:00 a 04:00 (UTC)
Europa (Irlanda)	De 22:00 a 06:00 (UTC)
Europa (Londres)	De 22:00 a 06:00 (UTC)
Europa (Estocolmo)	23h às 7h (UTC)
Europa (Milão)	De 21:00 a 05:00 (UTC)
Europa (Espanha)	De 21:00 a 05:00 (UTC)
África (Cidade do Cabo)	De 20:00 a 04:00 (UTC)
Europa (Irlanda)	De 22:00 a 06:00 (UTC)
Europa (Londres)	23h às 7h (UTC)
Europa (Paris)	23h às 7h (UTC)
Europa (Estocolmo)	23h às 7h (UTC)
Oriente Médio (Bahrein)	De 13h às 21h (UTC)
Oriente Médio (Emirados Árabes Unidos)	Das 18h às 2h (UTC)

Região	Janela de tempo de manutenção
América do Sul (São Paulo)	Das 19h às 3h (UTC)
Israel (Tel Aviv)	De 20:00 a 04:00 (UTC)

Defina um UUID para todos os operadores

Quando o Managed Service for Apache Flink inicia um trabalho do Flink para um aplicativo com um snapshot, o trabalho do Flink pode falhar ao iniciar devido a certos problemas. Um deles é a incompatibilidade de IDs do operador. O Flink espera IDs de operador explícitos e consistentes para operadores de gráficos de trabalhos do Flink. Se não for definido explicitamente, o Flink gera automaticamente um ID para os operadores. Isso ocorre porque o Flink usa esses IDs de operadores para identificar exclusivamente os operadores em um gráfico de trabalhos e os usa para armazenar o estado de cada operador em um ponto de salvamento.

O problema de incompatibilidade de IDs do operador ocorre quando o Flink não encontra um mapeamento 1:1 entre os IDs do operador de um gráfico de tarefas e os IDs do operador definidos em um ponto de salvamento. Isso acontece quando IDs de operadores explícitos e consistentes não são definidos e o Flink gera automaticamente IDs de operadores que podem não ser consistentes com a criação de cada gráfico de trabalho. A probabilidade de os aplicativos enfrentarem esse problema é alta durante as operações de manutenção. Para evitar isso, recomendamos que os clientes definam o UUID para todos os operadores no código flink. Para obter mais informações, consulte o tópico Definir um UUID para todos os operadores em Pronto para a produção.

Prontidão de produção

Essa é uma coleção de aspectos importantes da execução de aplicativos de produção no Managed Service for Apache Flink. Não é uma lista exaustiva, mas sim o mínimo que você deve prestar atenção antes de colocar um aplicativo em produção.

Teste de carga de aplicações

Alguns problemas com aplicativos só se manifestam sob carga pesada. Vimos casos em que os aplicativos pareciam íntegros e um evento operacional amplificou substancialmente a carga sobre o aplicativo. Isso pode acontecer de forma totalmente independente do próprio aplicativo: se a fonte de dados ou o coletor de dados ficar indisponível por algumas horas, o aplicativo Flink não poderá progredir. Depois que o problema é corrigido, há um acúmulo de dados não processados que pode esgotar completamente os recursos disponíveis. A carga pode então amplificar bugs ou problemas de desempenho que não surgiram antes.

Portanto, é essencial executar testes de carga adequados para aplicativos de produção. As perguntas que devem ser respondidas durante esses testes de carga incluem:

- O aplicativo é estável sob alta carga sustentada?
- O aplicativo ainda pode salvar em um ponto de salvamento sob carga máxima?
- Quanto tempo leva para processar um backlog de 1 hora? E por quanto tempo durante 24 horas (dependendo da retenção máxima dos dados no fluxo)?
- O throughput do aplicativo aumenta quando o aplicativo é escalado?

Ao consumir de um fluxo de dados, esses cenários podem ser simulados produzindo no fluxo por algum tempo. Em seguida, inicie o aplicativo e faça com que ele consuma dados desde o início do tempo, por exemplo, use uma posição inicial de `TRIM_HORIZON` no caso de um fluxo de dados Kinesis.

Paralelismo máximo

O paralelismo máximo define o paralelismo máximo para o qual um aplicativo com estado pode ser escalado. Isso é definido quando o estado é criado pela primeira vez e não há como escalar o operador além desse máximo sem descartar o estado.

O paralelismo máximo é definido quando o estado é criado pela primeira vez.

Por padrão, o paralelismo máximo é definido como:

- 128, se o paralelismo for <= 128
- `MIN(nextPowerOfTwo(parallelism + (parallelism / 2)), 2^15)`: se o paralelismo for > 128

Se planeja escalar seu aplicativo com paralelismo > 128, você deve definir explicitamente o paralelismo máximo.

O paralelismo máximo pode ser definido no nível da aplicação, com `env.setMaxParallelism(x)` ou um único operador. A menos que especificado de outra forma, todos os operadores herdam o paralelismo máximo do aplicativo.

Para obter mais informações, consulte Definir um paralelismo máximo explícito na documentação do Flink.

Defina um UUID para todos os operadores

Um UUID é usado na operação na qual o Flink mapeia um ponto de salvamento de volta para um operador individual. Definir um UUID específico para cada operador fornece um mapeamento estável para a restauração do processo de ponto de salvamento.

```
.map(...).uid("my-map-function")
```

Para obter mais informações, consulte Lista de verificação de prontidão de produção.

Práticas recomendadas do Managed Service for Apache Flink

Esta seção contém informações e recomendações para o desenvolvimento de um aplicativo Managed Service for Apache Flink estável e de alto desempenho.

Tópicos

Tolerância a falhas: pontos de verificação e pontos de salvamento

Use os pontos de verificação e os pontos de salvamento para implementar a tolerância a falhas em seu aplicativo Managed Service for Apache Flink. Lembre-se disso ao desenvolver e manter seu aplicativo:

- Recomendamos deixar o ponto de verificação habilitado para o seu aplicativo. O ponto de verificação fornece tolerância a falhas para o seu aplicativo durante a manutenção programada, assim como no caso de falhas inesperadas devido a problemas de serviço, falhas de dependência do aplicativo e outros problemas. Para obter mais informações manutenções programas, consulte Manutenção.

- Defina ApplicationSnapshotConfiguration:: SnapshotsEnabled para `false` durante o desenvolvimento ou solução de problemas do aplicativo. Um snapshot é criado durante cada parada do aplicativo, o que pode causar problemas se o aplicativo não estiver íntegro ou não estiver funcionando. Defina `SnapshotsEnabled` para `true` depois que o aplicativo estiver em produção e estável.

  > **Note**
  >
  > Recomendamos que seu aplicativo crie um instantâneo várias vezes ao dia para reiniciar adequadamente com os dados de estado corretos. A frequência correta para seus instantâneos depende da lógica de negócios do seu aplicativo. Tirar snapshots com frequência permite recuperar dados mais recentes, mas aumenta os custos e exige mais recursos do sistema.

  Para obter informações sobre como monitorar o tempo de inatividade do aplicativo, consulte Métricas e dimensões no Managed Service for Apache Flink.

Para obter mais informações sobre a implementação da tolerância a falhas, consulte Tolerância a falhas.

Versões de conectores incompatíveis

A versão 1.15 do Managed Service for Apache Flink impedirá, automaticamente, que os aplicativos sejam iniciados ou atualizados se estiverem usando versões incompatíveis do Kinesis Connector (empacotadas nos JARs do aplicativo). Ao fazer o upgrade para a versão 1.15 do Managed Service for Apache Flink, verifique se você está usando o Kinesis Connector mais recente. Isso quer dizer, qualquer versão igual ou mais recente do que a versão 1.15.2. Todas as outras versões não serão compatíveis com o Managed Service for Apache Flink, pois podem causar problemas de consistência ou falhas com o recurso Stop with Savepoint (Parar com o ponto de salvamento), impedindo operações limpas de parada/atualização.

Desempenho e paralelismo

Seu aplicativo pode escalar para atender a qualquer nível de throughput ajustando o paralelismo do aplicativo e evitando problemas de desempenho. Lembre-se disso ao desenvolver e manter seu aplicativo:

- Verifique se todas as fontes e coletores do seu aplicativo estão suficientemente provisionados e não estão sendo limitados. Se as fontes e os coletores forem outros AWS serviços, monitore esses serviços usando CloudWatch.
- Para aplicativos com paralelismo muito alto, verifique se os altos níveis de paralelismo são aplicados a todos os operadores no aplicativo. Por padrão, o Apache Flink aplica o mesmo paralelismo de aplicativos para todos os operadores no gráfico do aplicativo. Isso pode causar problemas de provisionamento em fontes ou coletores ou gargalos no processamento de dados do operador. Você pode alterar o paralelismo de cada operador no código com setParallelism.
- Entenda o significado das definições do paralelismo para os operadores em seu aplicativo. Se você alterar o paralelismo de um operador, talvez você não consiga restaurar o aplicativo a partir de um snapshot criado quando o operador tinha um paralelismo incompatível com as configurações atuais. Para obter mais informações sobre como definir o paralelismo do operador, consulte Definir explicitamente o paralelismo máximo para os operadores.

Para obter mais informações sobre a implementação da escalabilidade, consulte Escalabilidade.

Definindo o paralelismo por operador

Por padrão, todos os operadores têm o paralelismo definido no nível do aplicativo. Você pode substituir o paralelismo de um único operador usando a API usando. DataStream `.setParallelism(x)` Você pode definir um paralelismo do operador para qualquer paralelismo igual ou inferior ao paralelismo do aplicativo.

Se possível, defina o paralelismo do operador em função do paralelismo do aplicativo. Dessa forma, o paralelismo do operador variará com o paralelismo do aplicativo. Se você estiver usando o ajuste de escala automático, por exemplo, todos os operadores irão variar seu paralelismo na mesma proporção:

```
int appParallelism = env.getParallelism();
...
...ops.setParalleism(appParallelism/2);
```

Em alguns casos, você pode querer definir o paralelismo do operador como uma constante. Por exemplo, definir o paralelismo de uma fonte do Kinesis Stream para o número de fragmentos. Nesses casos, considere passar o paralelismo do operador como parâmetro de configuração do aplicativo, a fim de alterá-lo sem alterar o código, se você precisar, por exemplo, refragmentar o fluxo de origem.

Registro em log

Você pode monitorar o desempenho e as condições de erro do seu aplicativo usando o CloudWatch Logs. Lembre-se disso ao configurar o log para o aplicativo:

- Ative o CloudWatch registro do aplicativo para que quaisquer problemas de tempo de execução possam ser depurados.
- Não crie uma entrada de log para cada registro que está sendo processado no aplicativo. Isso causa gargalos graves durante o processamento e pode levar à contrapressão no processamento de dados.
- Crie CloudWatch alarmes para notificá-lo quando seu aplicativo não estiver funcionando corretamente. Para obter mais informações, consulte Alarmes.

Para obter mais informações sobre o registro em log, consulte Registro e Monitoramento.

Codificação

Você pode tornar seu aplicativo eficiente e estável usando as práticas de programação recomendadas. Lembre-se disso ao escrever o código do aplicativo:

- Não use `system.exit()` no código do aplicativo, no `main` método do aplicativo ou em funções definidas pelo usuário. Se você quiser desligar seu aplicativo a partir do código, lance uma exceção derivada de `Exception` ou `RuntimeException` contendo uma mensagem sobre o que deu errado com o aplicativo.

  Observe a seguir como o serviço lida com essa exceção:

  - Se a exceção for gerada pelo método `main` do seu aplicativo, o serviço a encapsulará em um `ProgramInvocationException` quando o aplicativo fizer a transição para o status `RUNNING`, e o Job Manager não enviará a tarefa.
  - Se a exceção for lançada a partir de uma função definida pelo usuário, o Job Manager falhará a tarefa e a reiniciará, e os detalhes da exceção serão gravados no log de exceções.
- Considere sombrear o arquivo JAR do aplicativo e suas dependências incluídas. O sombreamento é recomendado quando há possíveis conflitos nos nomes dos pacotes entre seu aplicativo e o runtime do Apache Flink. Se ocorrer um conflito, os registros do seu aplicativo podem conter uma exceção do tipo `java.util.concurrent.ExecutionException`. Para obter mais

informações sobre como sombrear o arquivo JAR do aplicativo, consulte Apache Maven Shade Plugin.

Gerenciamento de credenciais

Você não deve incorporar nenhuma credencial de longo prazo em aplicativos de produção (ou em qualquer outro). As credenciais de longo prazo são, provavelmente, verificadas em um sistema de controle de versão e podem ser facilmente perdidas. Em vez disso, você pode associar um perfil ao aplicativo Managed Service for Apache Flink e conceder privilégios a esse perfil. O aplicativo Flink em execução pode, em seguida, obter credenciais temporárias com os respectivos privilégios do ambiente. Caso seja necessário autenticar um serviço que não está nativamente integrado ao IAM, por exemplo, um banco de dados que exige um nome de usuário e senha para autenticação, considere armazenar segredos no AWSSecrets Manager.

Muitos serviços nativos AWS oferecem suporte à autenticação:

- Kinesis Data ProcessTaxiStream Streams — .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 imediatamente no Managed Service para Apache Flink

Lendo a partir de fontes com poucos fragmentos/partições

Ao se ler a partir do Apache Kafka ou de um Kinesis Data Stream, pode haver uma incompatibilidade entre o paralelismo do fluxo (ou seja, o número de partições do Kafka e o número de fragmentos do Kinesis) e o paralelismo do aplicativo. Com um design simples, o paralelismo de um aplicativo não pode escalar além do paralelismo de um fluxo: cada subtarefa de um operador de origem só pode ler a partir de um ou mais fragmentos/partições. Isso significa que, para um fluxo com apenas dois fragmentos e um aplicativo com um paralelismo de oito, apenas duas subtarefas estão realmente consumindo o fluxo e seis subtarefas permanecem inativas. Isso pode limitar substancialmente o throughput do aplicativo, especialmente se a desserialização for cara e realizada pela fonte (que é o padrão).

Para mitigar esse efeito, você pode escalar o fluxo. Mas, isso nem sempre é desejável ou possível. Como alternativa, você pode reestruturar a fonte para que ela não faça nenhuma serialização e só

transmita o `byte[]`. Em seguida, você pode reequilibrar os dados para distribuí-los uniformemente entre todas as tarefas e, em seguida, desserializar os dados lá. Dessa forma, você pode aproveitar todas as subtarefas para a desserialização e essa operação potencialmente cara não estará mais limitada ao número de fragmentos/partições do fluxo.

Intervalo de atualização de um notebook com Studio

Se você alterar o parágrafo de resultado do intervalo de atualização, defina-o para um valor de pelo menos 1.000 milissegundos.

Desempenho ideal de um notebook com Studio

Testamos com a seguinte instrução e obtivemos o melhor desempenho quando `events-per-second` multiplicado por `number-of-keys` menos de 25.000.000. Isso foi para `events-per-second` abaixo de 150.000.

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

Como as estratégias de marca d'água e os fragmentos inativos afetam as janelas de tempo

Ao ler os eventos do Apache Kafka e do Kinesis Data Streams, a fonte pode definir o horário do evento com base nos atributos do fluxo. No caso do Kinesis, o horário do evento é igual ao horário aproximado da chegada dos eventos. Mas, definir o horário do evento na origem dos eventos não é suficiente para que um aplicativo Flink use o horário do evento. A fonte também deve gerar marcas d'água que propaguem informações sobre o horário do evento da fonte para todos os outros operadores. A documentação do Flink apresenta uma visão geral abrangente sobre como esse processo funciona.

Por padrão, o registro de data e horário de um evento lido do Kinesis é definido como o horário aproximado de chegada determinado pelo Kinesis. Um pré-requisito adicional para que o horário do evento funcione no aplicativo é uma estratégia de marca d'água.

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

A estratégia de marcas d'água é então aplicada a um `DataStream` com o método `assignTimestampsAndWatermarks`. Existem algumas estratégias integradas úteis:

- `forMonotonousTimestamps()` usará apenas o horário do evento (hora aproximada de chegada) e emitirá periodicamente o valor máximo como marca d'água (para cada subtarefa específica)
- `forBoundedOutOfOrderness(Duration.ofSeconds(...))` semelhante à estratégia anterior, mas usará o tempo – duração do evento para gerar a marca d'água.

Isso funciona, mas há algumas ressalvas que você deve observar. As marcas d'água são geradas em um nível de subtarefa e fluem através do gráfico do operador.

Da documentação do Flink:

Geralmente, cada subtarefa paralela de uma função de origem gera suas marcas d'água de forma independente. Essas marcas d'água definem o horário do evento nessa fonte paralela específica.

Conforme as marcas d'água fluem pelo programa de streaming, elas avançam o horário do evento nos operadores onde chegam. Sempre que um operador avança o horário do evento, ele gera uma nova marca d'água posterior para seus operadores sucessores.

Alguns operadores consomem vários fluxos de entrada; uma união, por exemplo, ou operadores que seguem uma função keyBy (…) ou partition(…). O horário atual do evento desse operador é o tempo mínimo dos eventos de seus fluxos de entrada. À medida que seus fluxos de entrada atualizam os horários dos eventos, o mesmo acontece com o operador.

Isso significa que, se uma subtarefa de origem estiver consumindo um fragmento inativo, os operadores posteriores não recebem novas marcas d'água dessa subtarefa e, portanto, o processamento é interrompido para todos os operadores posteriores que usam janelas de tempo. Para evitar isso, os clientes podem adicionar a opção `withIdleness` à estratégia de marcas d'água. Com essa opção, um operador exclui as marcas d'água das subtarefas inativas anteriores ao calcular o horário do evento do operador. Portanto, a subtarefa inativa não bloqueia mais o avanço do horário do evento nos operadores posteriores.

No entanto, a opção de inatividade com as estratégias integradas de marca d'água não avançará o horário do evento se nenhuma subtarefa estiver lendo nenhum evento, ou seja, se não houver eventos no fluxo. Isso se torna particularmente visível em casos de teste em que um conjunto finito de eventos é lido a partir do fluxo. Como a hora do evento não avança após a leitura do último evento, a última janela (que contém o último evento) nunca será fechada.

Resumo

- a configuração `withIdleness` não gerará novas marcas d'água caso algum fragmento fique inativo, ela apenas excluirá a última marca d'água enviada por subtarefas inativas do cálculo mínimo de marcas d'água nos operadores posteriores
- com as estratégias de marca d'água incorporadas, a última janela aberta nunca será fechada (a menos que novos eventos que avancem a marca d'água sejam enviados, mas isso cria uma nova janela que permanece aberta)
- mesmo quando a hora é definida pelo fluxo do Kinesis, eventos de chegada tardia ainda podem ocorrer se um fragmento for consumido mais rápido do que outros (por exemplo, durante a inicialização do aplicativo ou ao usar `TRIM_HORIZON` onde todos os fragmentos existentes são consumidos paralelamente, ignorando a relação pai/filho)
- as configurações `withIdleness` da estratégia de marcas d'água parecem descontinuar as configurações específicas da fonte do Kinesis para fragmentos inativos (`ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`

Exemplo

O aplicativo a seguir está lendo a partir de um fluxo e criando janelas de sessão com base no horário do 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(l -> 0l)
    .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("XXXXXXXXXXXXXX Window with " + count + " events");
            System.out.println("; Watermark: " + timestamp + ", " +
 Instant.ofEpochMilli(timestamp));


            for (Long l : iterable) {
                System.out.println(l);
            }
        }
    });
```

No exemplo a seguir, oito eventos são gravados em um fluxo de 16 fragmentos (os dois primeiros e o último evento caem no mesmo fragmento).

```
$ 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 kinesis put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kinesis 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 kinesis 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 kinesis 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 kinesis 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
```

Essa entrada deve resultar em cinco janelas de sessão: evento 1, 2, 3; evento 4,5; evento 6; evento 7; evento 8. No entanto, o programa só produz as primeiras quatro janelas.

```
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:
 850705917302346158658436518579420528 64,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:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},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:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 496278943384362495989125660432414778027473288653837435 54,}}'} 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:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 496278943384139488537140354200999420844746805038777631 22,}}'} 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:
 212676479325586539664609129644855132 16,EndingHashKey:
 425352958651173079329218259289710264 31},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022 58,}}'}, 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:
 191408831393027885698148216680369618 944,EndingHashKey:
 212676479325586539664609129644855132 159},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457 14,}}'}, 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:
 212676479325586539664609129644855132 15},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218 26,}}'}, 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:
 255211775190703847597530955573826158 592,EndingHashKey:
 276479423123262501563991868538311671 807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870 10,}}'}, 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:
 496278943385254525797066885358076206758379223114076652 82,}}'}, 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:
 212676479325586539664609129644855132 16,EndingHashKey:
 425352958651173079329218259289710264 31},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022 58,}}'} 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:
 496278943385923548153022804052322278306558673959256065 78,}}'}, 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:
 212676479325586539664609129644855132 15},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218 26,}}'} 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:
 425352958651173079329218259289710264 32,EndingHashKey:
 638029437976759618993827388934565396 47},SequenceNumberRange: {StartingSequenceNumber:
 496278943383916481085155047969584063662020321423717826 90,}}'}, 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:
 496278943386369563056993416515152992672011641189375674 42,}}'}, 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:
 496278943385254525797066885358076206758379223114076652 82,}}'} 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:
 496278943386369563056993416515152992672011641189375674 42,}}'} 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:
 496278943386146555605008110283737635489285157574315870 10,}}'} 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:
 496278943385923548153022804052322278306558673959256065 78,}}'} 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
XXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
```

```
6
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030887170, 2022-03-23T10:21:27.170Z
7
```

A saída mostra apenas quatro janelas (faltando a última janela contendo o evento 8). Isso se deve ao horário do evento e à estratégia de marcas d'água. A última janela não pode ser fechada porque, com as estratégias de marcas d'água pré-criadas, o tempo nunca avança além do horário do último evento que foi lido a partir do fluxo. Mas, para que a janela se feche, o tempo precisa avançar mais de dez segundos após o último evento. Nesse caso, a última marca d'água é 2022-03-23T10:21:27.170Z, mas, para que a janela da sessão se feche, é necessária uma marca d'água de 10 segundos e 1 ms depois.

Se a opção `withIdleness` for removida da estratégia de marcas d'água, nenhuma janela da sessão será fechada, pois a "marca d'água global" do operador da janela não pode avançar.

Observe que, quando o aplicativo Flink é iniciado (ou se houver distorção de dados), alguns fragmentos podem ser consumidos mais rapidamente do que outros. Isso pode fazer com que algumas marcas d'água de uma subtarefa sejam emitidas muito cedo (a subtarefa pode emitir a marca d'água com base no conteúdo de um fragmento sem ter sido consumida dos outros fragmentos nos quais está inscrita). As formas de mitigar isso é uma estratégia diferente de marcas d'água que adicione um buffer de segurança (`forBoundedOutOfOrderness(Duration.ofSeconds(30))` ou permita explicitamente a chegada tardia de eventos (`allowedLateness(Time.minutes(5))`.

Defina um UUID para todos os operadores

Quando o Managed Service for Apache Flink inicia um trabalho do Flink para um aplicativo com um snapshot, o trabalho do Flink pode falhar ao iniciar devido a certos problemas. Um deles é a incompatibilidade de IDs do operador. O Flink espera IDs de operador explícitos e consistentes para operadores de gráficos de trabalhos do Flink. Se não for definido explicitamente, o Flink gera automaticamente um ID para os operadores. Isso ocorre porque o Flink usa esses IDs de operadores para identificar exclusivamente os operadores em um gráfico de trabalhos e os usa para armazenar o estado de cada operador em um ponto de salvamento.

O problema de incompatibilidade de IDs do operador ocorre quando o Flink não encontra um mapeamento 1:1 entre os IDs do operador de um gráfico de tarefas e os IDs do operador definidos em um ponto de salvamento. Isso acontece quando IDs de operadores explícitos e consistentes não são definidos e o Flink gera automaticamente IDs de operadores que podem não ser consistentes

com a criação de cada gráfico de trabalho. A probabilidade de os aplicativos enfrentarem esse problema é alta durante as operações de manutenção. Para evitar isso, recomendamos que os clientes definam o UUID para todos os operadores no código flink. Para obter mais informações, consulte o tópico Definir um UUID para todos os operadores em Pronto para a produção.

Adicionar ServiceResourceTransformer ao plug-in Maven Shade

O Flink usa as interfaces de provedores de serviços (Service Provider Interfaces, SPI) do Java para carregar componentes como conectores e formatos. Várias dependências do Flink que usam SPIs podem causar conflitos no uber-jar e comportamentos inesperados do aplicativo. É recomendável adicionar o ServiceResourceTransformerplugin maven shade, definido no 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>
```

Funções com estado no Apache Flink

Stateful Functions é uma API que simplifica a criação de aplicativos distribuídos com estado. É baseado em funções com estado persistente que podem interagir dinamicamente com fortes garantias de consistência.

Um aplicativo Stateful Functions é basicamente um aplicativo Apache Flink e, portanto, pode ser implantado no Managed Service for Apache Flink. No entanto, há algumas diferenças entre empacotar Stateful Functions para um cluster Kubernetes e para o Managed Service for Apache Flink. O aspecto mais importante de um aplicativo Stateful Functions é que a configuração do módulo contém todas as informações de runtime necessárias para configurar o runtime do Stateful Functions. Essa configuração geralmente é empacotada em um contêiner específico do Stateful Functions e implantada no Kubernetes. Mas isso não é possível com o Managed Service for Apache Flink.

A seguir está uma adaptação do exemplo StateFun Python para Managed Service para Apache Flink:

Modelo de aplicativo Apache Flink

Em vez de usar um contêiner de cliente para o runtime do Stateful Functions, os clientes podem compilar um jar de aplicativo Flink que apenas invoca o runtime do Stateful Functions e contém as dependências necessárias. Para o Flink 1.13, as dependências necessárias são semelhantes a estas:

```
<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>
```

E o método principal do aplicativo Flink para invocar o runtime do Stateful Function é assim:

```
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);
}
```

Observe que esses componentes são genéricos e independentes da lógica implementada no Stateful Function.

A localização da configuração do módulo

A configuração do módulo Stateful Functions precisa ser incluída no caminho da classe para ser descoberta no runtime do Stateful Functions. É melhor incluí-lo na pasta de recursos do aplicativo Flink e empacotá-lo no arquivo jar.

Semelhante a um aplicativo comum do Apache Flink, você pode usar o Maven em seguida para criar um arquivo uber jar e implantá-lo no Managed Service for Apache Flink.

Informações sobre a versão anterior do Managed Service for Apache Flink

Este tópico contém informações sobre como usar o Managed Service para Apache Flink com versões mais antigas do Apache Flink. As versões do Apache Flink compatíveis com o Managed Service for Apache Flink são 1.15.2 (recomendado), 1.13.2, 1.11.1, 1.8.2 e 1.6.2.

Recomendamos que você use a versão com suporte mais recente do Apache Flink com o seu aplicativo do Managed Service for Apache Flink. A versão 1.15.2 do Apache Flink tem os seguintes recursos:

- Suporte para API de tabela e SQL do Apache Flink
- Suporte para aplicativos Python.
- Suporte para Java versão 11 e qualquer versão do Scala
- Um modelo de memória aprimorado
- Otimizações do RocksDB para maior estabilidade do aplicativo
- Suporte para gerenciador de tarefas e rastreamentos de pilha no painel do Apache Flink.

Este tópico contém as seguintes seções:

- Usar o conector de fluxos Kinesis para o Apache Flink com versões anteriores do Apache Flink
- Construir aplicações com o Apache Flink 1.8.2
- Construir aplicações com o Apache Flink 1.6.2
- Atualização de aplicativos
- Conectores disponíveis no Apache Flink 1.6.2 e 1.8.2
- Introdução: Flink 1.13.2
- Introdução: Flink 1.11.1
- Introdução: Flink 1.8.2
- Introdução: Flink 1.6.2

Usar o conector de fluxos Kinesis para o Apache Flink com versões anteriores do Apache Flink

O conector de fluxos Kinesis para o Apache Flink não estava incluído no Apache Flink antes da versão 1.11. Para que seu aplicativo use o conector Kinesis para o Apache Flink com versões anteriores do Apache Flink, você deve baixar, compilar e instalar a versão do Apache Flink que seu aplicativo usa. Esse conector é usado para consumir dados de um Kinesis Stream usado como fonte do aplicativo ou para gravar dados em um Kinesis Stream usado para saída do aplicativo.

> **Note**
>
> Certifique-se de criar o conector com a versão 0.14.0 do KPL ou superior.

Para baixar e instalar o código-fonte do Apache Flink versão 1.8.2, faça o seguinte:

1. Certifique-se de ter o Apache Maven instalado e que sua variável de ambiente `JAVA_HOME` aponte para um JDK em vez de um JRE. Você pode testar a instalação do Apache Maven com o seguinte comando:

   ```
   mvn -version
   ```

2. Baixe o código-fonte do Apache Flink versão 1.8.2:

   ```
   wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
   ```

3. Descompacte o código-fonte do Apache Flink:

   ```
   tar -xvf flink-1.8.2-src.tgz
   ```

4. Vá para o diretório do código-fonte do Apache Flink:

   ```
   cd flink-1.8.2
   ```

5. Compile e instale o Apache Flink:

   ```
   mvn clean install -Pinclude-kinesis -DskipTests
   ```

> **Note**
>
> Se você estiver compilando o Flink no Microsoft Windows, precisará adicionar o parâmetro `-Drat.skip=true`.

Construir aplicações com o Apache Flink 1.8.2

Esta seção contém informações sobre os componentes que você usa para criar aplicativos do Managed Service para Apache Flink que funcionam com o Apache Flink 1.8.2.

Use as seguintes versões de componentes para os aplicativos do Managed Service for Apache Flink:

Componente	Versão
Java	1.8 (recomendado)
Apache Flink	1.8.2
Runtime do Managed Service for Apache Flink (aws-kinesisanalytics-runtime)	1.0.1
Conectores do Managed Service for Apache Flink (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1

Para compilar um aplicativo usando o Apache Flink 1.8.2, execute o Maven com o seguinte parâmetro:

```
mvn package -Dflink.version=1.8.2
```

Para obter um exemplo de arquivo `pom.xml` para um aplicativo do Managed Service for Apache Flink que usa o Apache Flink versão 1.8.2, consulte Aplicativo de conceitos básicos do Managed Service for Apache Flink 1.8.2.

Para obter informações sobre como criar e usar o código de aplicativo para um aplicativo do Managed Service for Apache Flink, consulte. Criar aplicativos

Construir aplicações com o Apache Flink 1.6.2

Esta seção contém informações sobre os componentes que você usa para criar aplicativos do Managed Service for Apache Flink que funcionam com o Apache Flink 1.6.2.

Use as seguintes versões de componentes para os aplicativos do Managed Service for Apache Flink:

Componente	Versão
Java	1.8 (recomendado)
AWS SDK do Java	1.11.379
Apache Flink	1.6.2
Runtime do Managed Service for Apache Flink (aws-kinesisanalytics-runtime)	1.0.1
Conectores do Managed Service for Apache Flink (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1
Apache Beam	Não é compatível com o Apache Flink 1.6.2.

> **Note**
>
> Ao usar o Runtime do Managed Service for Apache Flink versão 1.0.1, você especifica a versão do Apache Flink em seu arquivo `pom.xml` em vez de usar o parâmetro `-Dflink.version` ao compilar o código do aplicativo.

Para obter um exemplo de arquivo `pom.xml` para um aplicativo do Managed Service for Apache Flink que usa o Apache Flink versão 1.6.2, consulte Aplicativo de conceitos básicos do Managed Service for Apache Flink 1.6.2.

Para obter informações sobre como criar e usar o código de aplicativo para um aplicativo do Managed Service for Apache Flink, consulte. Criar aplicativos

Atualização de aplicativos

Para atualizar a versão de um aplicativo do Managed Service for Apache Flink, você deve atualizar o código do aplicativo, excluir o aplicativo anterior e criar um novo aplicativo com o código atualizado. Para isso, faça o seguinte:

- Altere as versões do runtime do Managed Service for Apache Flink e dos conectores do Managed Service for Apache Flink (aws-kinesisanalytics-flink) no arquivo `pom.xml` do seu aplicativo para 1.1.0.
- Remova a propriedade `flink.version` do arquivo `pom.xml` do seu aplicativo. Você fornecerá esse parâmetro ao compilar o código do aplicativo na próxima etapa.
- Recompile o código do seu aplicativo usando o seguinte comando:

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

- Exclua seu aplicativo existente. Crie seu aplicativo novamente e escolha Apache Flink versão 1.15.2 (versão recomendada) para o Runtime do aplicativo.

> **Note**
>
> Você não pode usar snapshots das versões anteriores do aplicativo.

Conectores disponíveis no Apache Flink 1.6.2 e 1.8.2

A estrutura do Apache Flink contém conectores para acessar dados de várias fontes.

- Para obter informações sobre conectores disponíveis na estrutura do Apache Flink 1.6.2, consulte Conectores (1.6.2) na Documentação do Apache Flink (1.6.2).
- Para obter informações sobre conectores disponíveis na estrutura do Apache Flink 1.8.2, consulte Conectores (1.8.2) na Documentação do Apache Flink (1.8.2).

Introdução: Flink 1.13.2

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink e da API. DataStream Ela descreve as opções disponíveis para criar e testar seus aplicativos. Ela também

fornece instruções para instalar as ferramentas necessárias para concluir os tutoriais deste guia e criar seu primeiro aplicativo.

Tópicos

Componentes de um aplicativo Managed Service for Apache Flink

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java/Apache Maven ou Scala que processa a entrada e produz a saída usando o runtime do Apache Flink.

O aplicativo do Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.
- Fonte: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um bucket do Amazon S3 etc. Para ter mais informações, consulte Origens.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Operadores da API do DataStream.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis Data Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Coletores.

Depois de criar, compilar e empacotar o código da aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon Simple Storage Service (Amazon S3). Em seguida, crie um

aplicativo do Managed Service for Apache Flink. Você passa na localização do pacote de código, um fluxo de dados do Kinesis como fonte de dados de fluxo e, normalmente, um local de fluxo ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos para concluir os exercícios

Para concluir as etapas neste guia, você deve ter o seguinte:

- Java Development Kit (JDK) versão 11. Defina a variável do ambiente`JAVA_HOME` para apontar para o local de instalação do JDK.
- Recomendamos que você use um ambiente de desenvolvimento (como Eclipse Java Neon ou IntelliJ Idea) para desenvolver e compilar seu aplicativo.
- Cliente do Git. Instale o cliente do Git se você ainda não tiver feito isso.
- Apache Maven Compiler Plugin. Maven deve estar em seu caminho de trabalho. Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

Para começar a usar, vá até Etapa 1: Configurar uma conta da AWS e criar um usuário administrador.

Etapa 1: Configurar uma conta da AWS e criar um usuário administrador

Cadastrar-se em uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas a seguir para criar uma.

Para se cadastrar em uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

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

   Quando você se cadastra em uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática

recomendada de segurança, atribua acesso administrativo a um usuário administrativo e use somente o usuário raiz para realizar as tarefas que exigem acesso do usuário raiz.

A AWS envia um e-mail de confirmação depois que o processo de cadastramento é concluído. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário administrativo

Depois de se inscrever em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o AWS IAM Identity Center e crie um usuário administrativo para não usar o usuário raiz em tarefas cotidianas.

Proteger seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Console como o proprietário da conta ao escolher a opção Usuário raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, digite sua senha.

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

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

Criar um usuário administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Enabling AWS IAM Identity Center no Manual do Usuário do AWS IAM Identity Center.
2. No Centro de Identidade do IAM, conceda acesso administrativo a um usuário administrativo.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configure user access with the default Diretório do Centro de Identidade do IAM no Manual do Usuário do AWS IAM Identity Center.

Login como usuário administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda com o login utilizando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Conceder acesso programático

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

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

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Configuração da AWS CLI para usar o AWS IAM Identity Center no AWS Command Line Interface Guia do usuário da . • Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações	Siga as instruções em Como usar credenciais temporári

Qual usuário precisa de acesso programático?	Para	Por
	programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	as com recursos da AWS no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Próxima etapa

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Próxima etapa

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura a AWS CLI para uso com o Managed Service for Apache Flink.

> **Note**
>
> Os exercícios de conceitos básicos neste guia pressupõem que você esteja usando credenciais de administrador (`adminuser`) em sua conta para executar as operações.

> **Note**
>
> Se você já tiver a AWS CLI instalada, pode ser necessário atualizá-la para obter as funcionalidades mais recentes. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface. Para verificar a versão da AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios neste tutorial requerem a seguinte versão da AWS CLI ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar a AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Guia do usuário do AWS Command Line Interface:
   - Instalar a AWS Command Line Interface
   - Configurar a AWS CLI
2. Adicione um perfil nomeado para o usuário administrador no arquivo AWS CLI `config`. Você pode usar esse perfil ao executar os comandos da AWS CLI. Para obter mais informações sobre perfis nomeados, consulte Perfis nomeados no Guia do usuário da 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 obter uma lista das regiões da AWS disponíveis, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

> **Note**
>
> O código e os comandos de exemplo neste tutorial usam a região Oeste dos EUA (Oregon). Para usar uma região diferente, altere a região no código e nos comandos deste tutorial para a região que você deseja usar.

3. Verifique a configuração digitando o seguinte comando no prompt de comando:

```
aws help
```

Depois de configurar uma AWS conta e aAWS CLI, você pode tentar o próximo exercício, no qual você configura um aplicativo de amostra e testa a end-to-end configuração.

Próxima etapa

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Neste exercício, você cria um aplicativo Managed Service for Apache Flink com fluxos de dados como origem e coletor.

Esta seção contém as seguintes etapas:

- Criar dois fluxos de dados do Amazon Kinesis Data Streams
- Gravação de registros de amostra no fluxo de entrada
- Baixar e examinar o código Java Apache Flink Streaming
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Próxima etapa

Criar dois fluxos de dados do Amazon Kinesis Data Streams

Antes de criar um aplicativo do Managed Service for Apache Flink para este exercício, crie dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`). O aplicativo usa esses fluxos para os fluxos de origem e de destino do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou o comando da AWS CLI a seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro fluxo (`ExampleInputStream`), use o comando `create-stream` AWS CLI do Amazon Kinesis a seguir.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Para criar o segundo fluxo que o aplicativo usa para gravar a saída, execute o mesmo comando, alterando o nome da transmissão para `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Mais adiante neste tutorial, você executa o script `stock.py` para enviar dados para o aplicativo.

```
$ python stock.py
```

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo Project Object Model (pom.xml) contém informações sobre a configuração e as dependências da aplicação, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.java` contém o método `main` que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Seu aplicativo cria conectores de origem e de destino para acessar recursos externos usando um objeto `StreamExecutionEnvironment`.
- O aplicativo cria conectores de origem e de destino usando propriedades estáticas. Para usar as propriedades do aplicativo dinâmico, use os métodos `createSinkFromApplicationProperties` e `createSourceFromApplicationProperties` para criar os conectores. Esses métodos leem as propriedades do aplicativo para configurar os conectores.

  Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compilar o código do aplicativo

Nesta seção, você usa o compilador do Apache Maven para criar o código Java para o aplicativo. Para obter informações sobre como instalar o Apache Maven e o Java Development Kit (JDK), consulte Pré-requisitos para concluir os exercícios.

Para compilar o código do aplicativo

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Há duas formas de compilar e empacotar o código:

   - Use a ferramenta de linha de comando do Maven. Crie seu arquivo JAR executando o seguinte comando no diretório que contém o arquivo `pom.xml`:

     ```
     mvn package -Dflink.version=1.13.2
     ```

- Use o ambiente de desenvolvimento. Consulte a documentação de seu ambiente de desenvolvimento para obter mais detalhes.

  > **Note**
  >
  > O código-fonte fornecido depende de bibliotecas do Java 11.

  Você pode carregar o pacote como um arquivo JAR, ou pode compactar o pacote e carregá-lo como um arquivo ZIP. Se você criar seu aplicativo usando a AWS CLI, especifique o tipo de conteúdo de código (JAR ou ZIP).

2. Se houver erros durante a compilação, verifique se sua variável de ambiente `JAVA_HOME` está definida corretamente.

Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

faça o upload do código Java Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon Simple Storage Service (Amazon S3) e faz upload do código do aplicativo.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira **ka-app-code-*<username>*** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).
5. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
6. Selecione Criar bucket.
7. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>

8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior. Selecione Next (Próximo).
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI.

> **Note**
>
> Quando você cria o aplicativo usando o console, seus recursos AWS Identity and Access Management (IAM) e do Amazon CloudWatch Logs são criados para você. Quando cria o aplicativo usando a AWS CLI, você cria esses recursos separadamente.

Tópicos

- Criar e executar o aplicativo (console)
- Criar e executar o aplicativo (AWS CLI)

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.

- Em Runtime, selecione Apache Flink.
- Deixe o menu suspenso de versão como Apache Flink versão 1.13.

4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-MyApplication-us-west-2`
> - Função: `kinesisanalytics-MyApplication-us-west-2`

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-west-2** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Insira o seguinte:

ID do grupo	Chave	Valor
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
6. Para CloudWatch registrar, marque a caixa de seleção Ativar.
7. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualize o aplicativo

Usando o console, você pode atualizar configurações do aplicativo, como as propriedades do aplicativo, as configurações de monitoramento e a localização ou o nome do arquivo JAR do aplicativo. Você também pode recarregar o JAR do aplicativo do bucket do Amazon S3 se precisar atualizar o código do aplicativo.

Na MyApplicationpágina, escolha Configurar. Atualize as configurações do aplicativo e selecione Update (Atualizar).

Criar e executar o aplicativo (AWS CLI)

Nesta seção, você usa a AWS CLI para criar e executar o aplicativo Flink do Managed Service for Apache Flink. O Managed Service for Apache Flink usa o comando `kinesisanalyticsv2` AWS CLI para criar e interagir com aplicativos Managed Service for Apache Flink.

Criar uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação `read` no fluxo de origem, e outra que concede permissões para ações `write` no fluxo de destino. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua *`username`* pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos Nomes de recurso da Amazon (ARNs) (*`012345678901`*) pelo ID da conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

> **Note**
>
> Para acessar outros serviços da Amazon, você pode usar o AWS SDK for Java. O Managed Service for Apache Flink define automaticamente as credenciais exigidas pelo SDK como as credenciais do perfil do IAM associado a seu aplicativo. Não é necessária nenhuma etapa adicional.

Criar uma perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles (Funções) e Create Role (Criar função).
3. Em Selecionar tipo de identidade de confiança, selecione Serviço da AWS. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis. Em Select your use case (Selecionar seu caso de uso), selecione Kinesis Analytics.

   Selecione Next: Permissions (Próximo: permissões).
4. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
5. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.

6. Anexe a política de permissões à função.

> **Note**
>
> Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, the section called “Criar uma política de permissões”.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).
   b. Selecione Attach Policies.
   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).
   d. Escolha a ReadSourceStreamWriteSinkStream política AK e escolha Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo do Managed Service for Apache Flink

1. Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (*`username`*) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (*`012345678901`*) na função de execução do serviço pelo ID da conta.

```
{
    "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. Execute a ação `CreateApplication` com a solicitação anterior para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação `StartApplication` para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação `StopApplication` para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Execute a ação `StopApplication` com a seguinte solicitação para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consultethe section called “Configurando o log”.

Atualizar propriedades do ambiente

Nesta seção, você usa a ação `UpdateApplication` para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação UpdateApplicationAWS CLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (*<username>*) com o sufixo que você selecionou na seção the section called “Criar dois fluxos de dados do 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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
```

```
        }
    }
}
```

Próxima etapa

[Etapa 4: limpar os recursos AWS]

Etapa 4: limpar os recursos AWS

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial Introdução.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Excluir seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos
- Próxima etapa

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seus fluxos de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Próxima etapa

Etapa 5: próximas etapas

Etapa 5: próximas etapas

Agora que você criou e executou um aplicativo básico do Managed Service for Apache Flink, consulte os seguintes recursos para obter soluções mais avançadas de Managed Service for Apache Flink.

- A solução de dados de transmissão para o Amazon Kinesis da AWS: A solução de dados de transmissão para o Amazon Kinesis da AWS configura automaticamente os serviços da AWS necessários para capturar, armazenar, processar e entregar dados de transmissão com facilidade. A solução oferece várias opções para resolver casos de uso de dados de transmissão. A opção Managed Service for Apache Flink fornece um exemplo de ETL de end-to-end streaming que demonstra um aplicativo do mundo real que executa operações analíticas em dados simulados de táxi de Nova York. A solução configura todos os AWS recursos necessários, como funções e políticas do IAM, um CloudWatch painel e CloudWatch alarmes.
- Solução de transmissão de dados para o Amazon MSK da AWS: A solução de transmissão de dados para o Amazon MSK da AWS fornece modelos de AWS CloudFormation onde os dados fluem por produtores, armazenamento de transmissão, consumidores e destinos.
- Clickstream Lab com Apache Flink e Apache Kafka: um laboratório completo para casos de uso de clickstream usando Amazon Managed Streaming for Apache Kafka para armazenamento de transmissão e aplicativos Managed Service for Apache Flink for Apache Flink para processamento de fluxos.
- Workshop do Amazon Managed Service para Apache Flink: Neste workshop, você cria uma arquitetura de end-to-end streaming para ingerir, analisar e visualizar dados de streaming quase em tempo real. Você decidiu melhorar as operações de uma empresa de táxi na cidade de Nova York. Você analisa os dados de telemetria de uma frota de táxis na cidade de Nova York quase em tempo real para otimizar as operações da frota.
- Managed Service for Apache Flink: exemplos: Esta seção deste Guia do desenvolvedor fornece exemplos de como criar e trabalhar com aplicativos no Managed Service for Apache Flink. Eles incluem exemplos de código e step-by-step instruções para ajudá-lo a criar serviços gerenciados para aplicativos Apache Flink e testar seus resultados.
- Conheça o Flink: treinamento prático: Treinamento introdutório oficial do Apache Flink que ajuda você a começar a escrever ETL de transmissão escalável, análises e aplicativos orientados a eventos.

> **Note**
>
> Esteja ciente de que o Managed Service for Apache Flink não é compatível com a versão Apache Flink (1.12) usada neste treinamento. Você pode usar o Flink 1.15.2 no Flink Managed Service para Apache Flink.

Introdução: Flink 1.11.1

Este tópico contém uma versão do Introdução (DataStream API) Tutorial que usa o Apache Flink 1.11.1.

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink e da API. DataStream Ela descreve as opções disponíveis para criar e testar seus aplicativos. Ela também fornece instruções para instalar as ferramentas necessárias para concluir os tutoriais deste guia e criar seu primeiro aplicativo.

Tópicos

Componentes de um aplicativo Managed Service for Apache Flink

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java/ Apache Maven ou Scala que processa a entrada e produz a saída usando o runtime do Apache Flink.

O Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.
- Fonte: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um bucket do Amazon S3 etc. Para ter mais informações, consulte Origens.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Operadores da API do DataStream.

- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis Data Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Coletores.

Depois de criar, compilar e empacotar o código da aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon Simple Storage Service (Amazon S3). Em seguida, crie um aplicativo do Managed Service for Apache Flink. Você passa na localização do pacote de código, um fluxo de dados do Kinesis como fonte de dados de fluxo e, normalmente, um local de fluxo ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos para concluir os exercícios

Para concluir as etapas neste guia, você deve ter o seguinte:

- Java Development Kit (JDK) versão 11. Defina a variável do ambiente`JAVA_HOME` para apontar para o local de instalação do JDK.
- Recomendamos que você use um ambiente de desenvolvimento (como Eclipse Java Neon ou IntelliJ Idea) para desenvolver e compilar seu aplicativo.
- Cliente do Git. Instale o cliente do Git se você ainda não tiver feito isso.
- Apache Maven Compiler Plugin. Maven deve estar em seu caminho de trabalho. Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

Para começar a usar, vá até Etapa 1: Configurar uma conta da AWS e criar um usuário administrador.

Etapa 1: Configurar uma conta da AWS e criar um usuário administrador

Cadastrar-se em uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas a seguir para criar uma.

Para se cadastrar em uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

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

Quando você se cadastra em uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua acesso administrativo a um usuário administrativo e use somente o usuário raiz para realizar as tarefas que exigem acesso do usuário raiz.

A AWS envia um e-mail de confirmação depois que o processo de cadastramento é concluído. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário administrativo

Depois de se inscrever em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o AWS IAM Identity Center e crie um usuário administrativo para não usar o usuário raiz em tarefas cotidianas.

Proteger seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Console como o proprietário da conta ao escolher a opção Usuário raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, digite sua senha.

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

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

Criar um usuário administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Enabling AWS IAM Identity Center no Manual do Usuário do AWS IAM Identity Center.
2. No Centro de Identidade do IAM, conceda acesso administrativo a um usuário administrativo.

Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configure user access with the default Diretório do Centro de Identidade do IAM no Manual do Usuário do AWS IAM Identity Center.

Login como usuário administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda com o login utilizando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Conceder acesso programático

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

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

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Configuração da AWS CLI para usar o AWS IAM Identity Center no AWS Command Line Interface Guia do usuário da . • Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência

Qual usuário precisa de acesso programático?	Para	Por
		de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções em Como usar credenciais temporárias com recursos da AWS no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Próxima etapa

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura a AWS CLI para uso com o Managed Service for Apache Flink.

> **Note**
>
> Os exercícios de conceitos básicos neste guia pressupõem que você esteja usando credenciais de administrador (`adminuser`) em sua conta para executar as operações.

> **Note**
>
> Se você já tiver a AWS CLI instalada, pode ser necessário atualizá-la para obter as funcionalidades mais recentes. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface. Para verificar a versão da AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios neste tutorial requerem a seguinte versão da AWS CLI ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar a AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Guia do usuário do AWS Command Line Interface:
   - Instalar a AWS Command Line Interface
   - Configurar a AWS CLI
2. Adicione um perfil nomeado para o usuário administrador no arquivo AWS CLI `config`. Você pode usar esse perfil ao executar os comandos da AWS CLI. Para obter mais informações sobre perfis nomeados, consulte Perfis nomeados no Guia do usuário da 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 obter uma lista das regiões da AWS disponíveis, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

> **Note**
>
> O código e os comandos de exemplo neste tutorial usam a região Oeste dos EUA (Oregon). Para usar uma região diferente, altere a região no código e nos comandos deste tutorial para a região que você deseja usar.

3. Verifique a configuração digitando o seguinte comando no prompt de comando:

```
aws help
```

Depois de configurar uma AWS conta e aAWS CLI, você pode tentar o próximo exercício, no qual você configura um aplicativo de amostra e testa a end-to-end configuração.

Próxima etapa

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Neste exercício, você cria um aplicativo Managed Service for Apache Flink com fluxos de dados como origem e coletor.

Esta seção contém as seguintes etapas:

Criar dois fluxos de dados do Amazon Kinesis Data Streams

Antes de criar um aplicativo do Managed Service for Apache Flink para este exercício, crie dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`). O aplicativo usa esses fluxos para os fluxos de origem e de destino do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou o comando da AWS CLI a seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro fluxo (`ExampleInputStream`), use o comando `create-stream` AWS CLI do Amazon Kinesis a seguir.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Para criar o segundo fluxo que o aplicativo usa para gravar a saída, execute o mesmo comando, alterando o nome da transmissão para `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Mais adiante neste tutorial, você executa o script `stock.py` para enviar dados para o aplicativo.

```
$ python stock.py
```

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo Project Object Model (pom.xml) contém informações sobre a configuração e as dependências da aplicação, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.java` contém o método `main` que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Seu aplicativo cria conectores de origem e de destino para acessar recursos externos usando um objeto `StreamExecutionEnvironment`.
- O aplicativo cria conectores de origem e de destino usando propriedades estáticas. Para usar as propriedades do aplicativo dinâmico, use os métodos `createSinkFromApplicationProperties` e `createSourceFromApplicationProperties` para criar os conectores. Esses métodos leem as propriedades do aplicativo para configurar os conectores.

  Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compilar o código do aplicativo

Nesta seção, você usa o compilador do Apache Maven para criar o código Java para o aplicativo. Para obter informações sobre como instalar o Apache Maven e o Java Development Kit (JDK), consulte Pré-requisitos para concluir os exercícios.

Para compilar o código do aplicativo

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Há duas formas de compilar e empacotar o código:

- Use a ferramenta de linha de comando do Maven. Crie seu arquivo JAR executando o seguinte comando no diretório que contém o arquivo `pom.xml`:

```
mvn package -Dflink.version=1.11.3
```

- Use o ambiente de desenvolvimento. Consulte a documentação de seu ambiente de desenvolvimento para obter mais detalhes.

> **Note**
>
> O código-fonte fornecido depende de bibliotecas do Java 11. Certifique-se de que a versão Java do seu projeto seja 11.

Você pode carregar o pacote como um arquivo JAR, ou pode compactar o pacote e carregá-lo como um arquivo ZIP. Se você criar seu aplicativo usando a AWS CLI, especifique o tipo de conteúdo de código (JAR ou ZIP).

2. Se houver erros durante a compilação, verifique se sua variável de ambiente `JAVA_HOME` está definida corretamente.

Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

faça o upload do código Java Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon Simple Storage Service (Amazon S3) e faz upload do código do aplicativo.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Insira **`ka-app-code-<username>`** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).

4. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).
5. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
6. Selecione Criar bucket.
7. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior. Selecione Next (Próximo).
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI.

> **Note**
>
> Quando você cria o aplicativo usando o console, seus recursos AWS Identity and Access Management (IAM) e do Amazon CloudWatch Logs são criados para você. Quando cria o aplicativo usando a AWS CLI, você cria esses recursos separadamente.

Tópicos

- Criar e executar o aplicativo (console)
- Criar e executar o aplicativo (AWS CLI)

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe o menu suspenso de versão como Apache Flink versão 1.11 (versão recomendada).
4. Em Permissões de acesso, escolha Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*`012345678901`*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, escolha Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Em Properties (Propriedades), Group ID (ID do grupo), insira **ProducerConfigProperties**.
5. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**

ID do grupo	Chave	Valor
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
7. Para CloudWatch registrar, marque a caixa de seleção Ativar.
8. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

O gráfico de tarefas do Flink pode ser visualizado executando o aplicativo, abrindo o painel do Apache Flink e selecionando a tarefa desejada do Flink.

Interromper o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualize o aplicativo

Usando o console, você pode atualizar configurações do aplicativo, como as propriedades do aplicativo, as configurações de monitoramento e a localização ou o nome do arquivo JAR do aplicativo. Você também pode recarregar o JAR do aplicativo do bucket do Amazon S3 se precisar atualizar o código do aplicativo.

Na MyApplicationpágina, escolha Configurar. Atualize as configurações do aplicativo e selecione Update (Atualizar).

Criar e executar o aplicativo (AWS CLI)

Nesta seção, você usa o AWS CLI para criar e executar o aplicativo Managed Service for Apache Flink. Um Managed Service for Apache Flink usa o `kinesisanalyticsv2` AWS CLI comando para criar e interagir com o Managed Service for Apache Flink.

Criar uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação `read` no fluxo de origem, e outra que concede permissões para ações `write` no fluxo de destino. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua *`username`* pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos Nomes de recurso da Amazon (ARNs) (*`012345678901`*) pelo ID da conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

> **Note**
>
> Para acessar outros serviços da Amazon, você pode usar o AWS SDK for Java. O Managed Service for Apache Flink define automaticamente as credenciais exigidas pelo SDK como as credenciais do perfil do IAM associado a seu aplicativo. Não é necessária nenhuma etapa adicional.

Criar uma perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles (Funções) e Create Role (Criar função).
3. Em Selecionar tipo de identidade de confiança, selecione Serviço da AWS. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis. Em Select your use case (Selecionar seu caso de uso), selecione Kinesis Analytics.

   Selecione Next: Permissions (Próximo: permissões).
4. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
5. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
6. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, the section called “Criar uma política de permissões”.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).

   b. Selecione Attach Policies.

   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).

   d. Escolha a ReadSourceStreamWriteSinkStream política AK e escolha Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo do Managed Service for Apache Flink

1. Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (*username*) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (*012345678901*) na função de execução do serviço pelo ID da conta.

```
{
    "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. Execute a ação `CreateApplication` com a solicitação anterior para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação `StartApplication` para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação `StopApplication` para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Execute a ação `StopApplication` com a seguinte solicitação para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consultethe section called “Configurando o log”.

Atualizar propriedades do ambiente

Nesta seção, você usa a ação `UpdateApplication` para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação `UpdateApplication`AWS CLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou

`ListApplications`. Atualize o sufixo do nome do bucket (*<username>*) com o sufixo que você selecionou na seção the section called “Criar dois fluxos de dados do 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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Próxima etapa

Etapa 4: limpar os recursos AWS

Etapa 4: limpar os recursos AWS

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial Introdução.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Excluir seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos
- Próxima etapa

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.

2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seus fluxos de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e, em seguida, confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.

4. Selecione Excluir grupo de logs e, em seguida, confirme a exclusão.

Próxima etapa

Etapa 5: próximas etapas

Etapa 5: próximas etapas

Agora que você criou e executou um aplicativo básico do Managed Service for Apache Flink, consulte os seguintes recursos para obter soluções mais avançadas de Managed Service for Apache Flink.

- A solução de dados de transmissão para o Amazon Kinesis da AWS: A solução de dados de transmissão para o Amazon Kinesis da AWS configura automaticamente os serviços da AWS necessários para capturar, armazenar, processar e entregar dados de transmissão com facilidade. A solução oferece várias opções para resolver casos de uso de dados de transmissão. A opção Managed Service for Apache Flink fornece um exemplo de ETL de end-to-end streaming que demonstra um aplicativo do mundo real que executa operações analíticas em dados simulados de táxi de Nova York. A solução configura todos os AWS recursos necessários, como funções e políticas do IAM, um CloudWatch painel e CloudWatch alarmes.
- Solução de transmissão de dados para o Amazon MSK da AWS: A solução de transmissão de dados para o Amazon MSK da AWS fornece modelos de AWS CloudFormation onde os dados fluem por produtores, armazenamento de transmissão, consumidores e destinos.
- Clickstream Lab com Apache Flink e Apache Kafka: um laboratório completo para casos de uso de clickstream usando Amazon Managed Streaming for Apache Kafka para armazenamento de transmissão e aplicativos Managed Service for Apache Flink for Apache Flink para processamento de fluxos.
- Workshop do Amazon Managed Service para Apache Flink: Neste workshop, você cria uma arquitetura de end-to-end streaming para ingerir, analisar e visualizar dados de streaming quase em tempo real. Você decidiu melhorar as operações de uma empresa de táxi na cidade de Nova York. Você analisa os dados de telemetria de uma frota de táxis na cidade de Nova York quase em tempo real para otimizar as operações da frota.
- Managed Service for Apache Flink: exemplos: Esta seção deste Guia do desenvolvedor fornece exemplos de como criar e trabalhar com aplicativos no Managed Service for Apache Flink. Eles incluem exemplos de código e step-by-step instruções para ajudar você a criar serviços gerenciados para aplicativos Apache Flink e testar seus resultados.

- Conheça o Flink: treinamento prático: Treinamento introdutório oficial do Apache Flink que ajuda você a começar a escrever ETL de transmissão escalável, análises e aplicativos orientados a eventos.

> **Note**
>
> Esteja ciente de que o Managed Service for Apache Flink não é compatível com a versão Apache Flink (1.12) usada neste treinamento. Você pode usar o Flink 1.15.2 no Flink Managed Service para Apache Flink.

- Exemplos de código do Apache Flink: um GitHub repositório de uma grande variedade de exemplos de aplicativos do Apache Flink.

Introdução: Flink 1.8.2

Este tópico contém uma versão do Tutorial Introdução (DataStream API) que usa o Apache Flink 1.8.2.

Tópicos

- Componentes do aplicativo do Managed Service for Apache Flink
- Pré-requisitos para concluir os exercícios
- Etapa 1: Configurar uma conta da AWS e criar um usuário administrador
- Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)
- Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink
- Etapa 4: limpar os recursos AWS

Componentes do aplicativo do Managed Service for Apache Flink

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java/Apache Maven ou Scala que processa a entrada e produz a saída usando o runtime do Apache Flink.

O Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.

- Fonte: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um bucket do Amazon S3 etc. Para ter mais informações, consulte Origens.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Operadores da API do DataStream.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis Data Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Coletores.

Depois de criar, compilar e empacotar o código da aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon Simple Storage Service (Amazon S3). Em seguida, crie um aplicativo do Managed Service for Apache Flink. Você passa na localização do pacote de código, um fluxo de dados do Kinesis como fonte de dados de fluxo e, normalmente, um local de fluxo ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos para concluir os exercícios

Para concluir as etapas neste guia, você deve ter o seguinte:

- Java Development Kit (JDK) versão 8. Defina a variável do ambiente`JAVA_HOME` para apontar para o local de instalação do JDK.
- Para usar o conector Apache Flink Kinesis neste tutorial, você deve baixar e instalar o Apache Flink. Para obter detalhes, consulte Usar o conector de fluxos Kinesis para o Apache Flink com versões anteriores do Apache Flink.
- Recomendamos que você use um ambiente de desenvolvimento (como Eclipse Java Neon ou IntelliJ Idea) para desenvolver e compilar seu aplicativo.
- Cliente do Git. Instale o cliente do Git se você ainda não tiver feito isso.
- Apache Maven Compiler Plugin. Maven deve estar em seu caminho de trabalho. Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

Para começar a usar, vá até Etapa 1: Configurar uma conta da AWS e criar um usuário administrador.

Etapa 1: Configurar uma conta da AWS e criar um usuário administrador

Cadastrar-se em uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas a seguir para criar uma.

Para se cadastrar em uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

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

   Quando você se cadastra em uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua acesso administrativo a um usuário administrativo e use somente o usuário raiz para realizar as tarefas que exigem acesso do usuário raiz.

A AWS envia um e-mail de confirmação depois que o processo de cadastramento é concluído. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário administrativo

Depois de se inscrever em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o AWS IAM Identity Center e crie um usuário administrativo para não usar o usuário raiz em tarefas cotidianas.

Proteger seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Console como o proprietário da conta ao escolher a opção Usuário raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, digite sua senha.

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

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

Criar um usuário administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Enabling AWS IAM Identity Center no Manual do Usuário do AWS IAM Identity Center.
2. No Centro de Identidade do IAM, conceda acesso administrativo a um usuário administrativo.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configure user access with the default Diretório do Centro de Identidade do IAM no Manual do Usuário do AWS IAM Identity Center.

Login como usuário administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda com o login utilizando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Conceder acesso programático

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

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

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho	Use credenciais temporárias para assinar solicitações programáticas para a AWS	Siga as instruções da interface que deseja utilizar.

Qual usuário precisa de acesso programático?	Para	Por
(Usuários gerenciados no Centro de Identidade do IAM)	CLI, os SDKs da AWS ou as APIs da AWS.	• Para a AWS CLI, consulte Configuração da AWS CLI para usar o AWS IAM Identity Center no AWS Command Line Interface Guia do usuário da . • Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções em Como usar credenciais temporárias com recursos da AWS no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura a AWS CLI para uso com o Managed Service for Apache Flink.

> **Note**
>
> Os exercícios de conceitos básicos neste guia pressupõem que você esteja usando credenciais de administrador (`adminuser`) em sua conta para executar as operações.

> **Note**
>
> Se você já tiver a AWS CLI instalada, pode ser necessário atualizá-la para obter as funcionalidades mais recentes. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface. Para verificar a versão da AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios neste tutorial requerem a seguinte versão da AWS CLI ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar a AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Guia do usuário do AWS Command Line Interface:
   - Instalar a AWS Command Line Interface
   - Configurar a AWS CLI
2. Adicione um perfil nomeado para o usuário administrador no arquivo AWS CLI `config`. Você pode usar esse perfil ao executar os comandos da AWS CLI. Para obter mais informações sobre perfis nomeados, consulte Perfis nomeados no Guia do usuário da 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 obter uma lista das regiões disponíveis, consulte Regiões e endpoints do na Referência geral da Amazon Web Services.

> **Note**
>
> O código e os comandos de exemplo neste tutorial usam a região Oeste dos EUA (Oregon). Para usar uma região AWS diferente, altere a região no código e nos comandos deste tutorial para a região que você deseja usar.

3. Verifique a configuração digitando o seguinte comando no prompt de comando:

```
aws help
```

Depois de configurar uma AWS conta e aAWS CLI, você pode tentar o próximo exercício, no qual você configura um aplicativo de amostra e testa a end-to-end configuração.

Próxima etapa

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Neste exercício, você cria um aplicativo Managed Service for Apache Flink com fluxos de dados como origem e coletor.

Esta seção contém as seguintes etapas:

- Criar dois fluxos de dados do Amazon Kinesis Data Streams
- Gravação de registros de amostra no fluxo de entrada
- Baixar e examinar o código Java Apache Flink Streaming
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink
- Próxima etapa

Criar dois fluxos de dados do Amazon Kinesis Data Streams

Antes de criar um aplicativo do Managed Service for Apache Flink para este exercício, crie dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`). O aplicativo usa esses fluxos para os fluxos de origem e de destino do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou o comando da AWS CLI a seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro fluxo (`ExampleInputStream`), use o comando `create-stream` AWS CLI do Amazon Kinesis a seguir.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Para criar o segundo fluxo que o aplicativo usa para gravar a saída, execute o mesmo comando, alterando o nome da transmissão para `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Mais adiante neste tutorial, você executa o script `stock.py` para enviar dados para o aplicativo.

```
$ python stock.py
```

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo Project Object Model (pom.xml) contém informações sobre a configuração e as dependências da aplicação, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.java` contém o método `main` que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Seu aplicativo cria conectores de origem e de destino para acessar recursos externos usando um objeto `StreamExecutionEnvironment`.
- O aplicativo cria conectores de origem e de destino usando propriedades estáticas. Para usar as propriedades do aplicativo dinâmico, use os métodos `createSinkFromApplicationProperties` e `createSourceFromApplicationProperties` para criar os conectores. Esses métodos leem as propriedades do aplicativo para configurar os conectores.

  Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compilar o código do aplicativo

Nesta seção, você usa o compilador do Apache Maven para criar o código Java para o aplicativo. Para obter informações sobre como instalar o Apache Maven e o Java Development Kit (JDK), consulte Pré-requisitos para concluir os exercícios.

> **Note**
>
> Para usar o conector Kinesis com versões do Apache Flink anteriores à 1.11, você precisa baixar, compilar e instalar o Apache Maven. Para obter mais informações, consulte the section called “Usar o conector de fluxos Kinesis para o Apache Flink com versões anteriores do Apache Flink”.

Para compilar o código do aplicativo

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Há duas formas de compilar e empacotar o código:

   - Use a ferramenta de linha de comando do Maven. Crie seu arquivo JAR executando o seguinte comando no diretório que contém o arquivo `pom.xml`:

     ```
     mvn package -Dflink.version=1.8.2
     ```

   - Use o ambiente de desenvolvimento. Consulte a documentação de seu ambiente de desenvolvimento para obter mais detalhes.

     > **Note**
     >
     > O código-fonte fornecido depende de bibliotecas do Java 1.8. Certifique-se de que a versão Java do seu projeto seja 1.8.

   Você pode carregar o pacote como um arquivo JAR, ou pode compactar o pacote e carregá-lo como um arquivo ZIP. Se você criar seu aplicativo usando a AWS CLI, especifique o tipo de conteúdo de código (JAR ou ZIP).

2. Se houver erros durante a compilação, verifique se sua variável de ambiente `JAVA_HOME` está definida corretamente.

Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

faça o upload do código Java Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon Simple Storage Service (Amazon S3) e faz upload do código do aplicativo.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.

3. Insira **ka-app-code-*<username>*** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).
5. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
6. Selecione Criar bucket.
7. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior. Selecione Next (Próximo).
9. Você não precisa alterar nenhuma das configurações para o objeto, em seguida, selecione Upload.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pelo aplicativo.

Criar e executar o aplicativo do Managed Service for Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI.

> **Note**
>
> Quando você cria o aplicativo usando o console, seus recursos AWS Identity and Access Management (IAM) e do Amazon CloudWatch Logs são criados para você. Quando cria o aplicativo usando a AWS CLI, você cria esses recursos separadamente.

Tópicos

- Criar e executar o aplicativo (console)
- Criar e executar o aplicativo (AWS CLI)

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.
   - Deixe o menu suspenso da versão como Apache Flink 1.8 (versão recomendada).
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.

3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
6. Para CloudWatch registrar, marque a caixa de seleção Ativar.
7. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

1. Na MyApplicationpágina, escolha Executar. Confirme a ação.
2. Quando o aplicativo estiver em execução, atualize a página. O console mostra o Gráfico do aplicativo.

Interromper o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualize o aplicativo

Usando o console, você pode atualizar configurações do aplicativo, como as propriedades do aplicativo, as configurações de monitoramento e a localização ou o nome do arquivo JAR do aplicativo. Você também pode recarregar o JAR do aplicativo do bucket do Amazon S3 se precisar atualizar o código do aplicativo.

Na MyApplicationpágina, escolha Configurar. Atualize as configurações do aplicativo e selecione Update (Atualizar).

Criar e executar o aplicativo (AWS CLI)

Nesta seção, você usa a AWS CLI para criar e executar o aplicativo Flink do Managed Service for Apache Flink. O Managed Service for Apache Flink usa o comando `kinesisanalyticsv2` AWS CLI para criar e interagir com aplicativos Managed Service for Apache Flink.

Criar uma política de permissões

> **Note**
>
> Você deve criar uma política de permissões e uma função para o seu aplicativo. Se você não criar esses recursos do IAM, seu aplicativo não poderá acessar seus fluxos de logs e dados.

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação `read` no fluxo de origem, e outra que concede permissões para ações `write` no fluxo de destino. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua *`username`* pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos Nomes de recurso da Amazon (ARNs) (*`012345678901`*) pelo ID da conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

> **Note**
>
> Para acessar outros serviços da Amazon, você pode usar o AWS SDK for Java. O Managed Service for Apache Flink define automaticamente as credenciais exigidas pelo SDK como as credenciais do perfil do IAM associado a seu aplicativo. Não é necessária nenhuma etapa adicional.

Criar uma perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles (Funções) e Create Role (Criar função).
3. Em Selecionar tipo de identidade de confiança, selecione Serviço da AWS. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis. Em Select your use case (Selecionar seu caso de uso), selecione Kinesis Analytics.

   Selecione Next: Permissions (Próximo: permissões).
4. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
5. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
6. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, the section called “Criar uma política de permissões”.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).

   b. Selecione Attach Policies.

c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).

d. Escolha a ReadSourceStreamWriteSinkStream política AK e escolha Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo do Managed Service for Apache Flink

1. Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (*`username`*) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (*`012345678901`*) na função de execução do serviço pelo ID da conta.

```
{
    "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. Execute a ação `CreateApplication` com a solicitação anterior para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação `StartApplication` para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação `StopApplication` para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Execute a ação `StopApplication` com a seguinte solicitação para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consultethe section called “Configurando o log”.

Atualizar propriedades do ambiente

Nesta seção, você usa a ação `UpdateApplication` para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação `UpdateApplication`AWS CLI.

> **Note**
>
> Para carregar uma nova versão do código do aplicativo com o mesmo nome de arquivo, você deve especificar a nova versão do objeto. Para obter mais informações sobre o uso de versões de objetos do Amazon S3, consulte Como ativar ou desativar o controle de versão.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3 e a nova versão do objeto. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (*<username>*) com o sufixo que você selecionou na seção the section called “Criar dois fluxos de dados do 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": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Próxima etapa

Etapa 4: limpar os recursos AWS

Etapa 4: limpar os recursos AWS

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial Introdução.

Este tópico contém as seguintes seções:

- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Selecione Configurar.
4. Na seção Snapshots, selecione Desativar e, em seguida, selecione Atualizar.
5. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seus fluxos de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.

8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e confirme a exclusão.

Introdução: Flink 1.6.2

Este tópico contém uma versão do Tutorial Introdução (DataStream API) que usa o Apache Flink 1.6.2.

Tópicos

- Componentes do Managed Service for Apache Flink
- Pré-requisitos para concluir os exercícios
- Etapa 1: Configurar uma conta da AWS e criar um usuário administrador
- Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)
- Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink
- Etapa 4: limpar os recursos AWS

Componentes do Managed Service for Apache Flink

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java/Apache Maven ou Scala que processa a entrada e produz a saída usando o runtime do Apache Flink.

um Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de runtime: você pode usar as propriedades de runtime para configurar seu aplicativo sem recompilar o código do aplicativo.
- Fonte: o aplicativo consome dados usando uma fonte. Um conector de origem lê dados de um fluxo de dados do Kinesis, de um bucket do Amazon S3 etc. Para ter mais informações, consulte Origens.

- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Operadores da API do DataStream.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector do coletor grava dados em um fluxo de dados do Kinesis Data Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Coletores.

Depois de criar, compilar e empacotar o seu aplicativo, é necessário fazer o upload do pacote do código em um bucket do Amazon Simple Storage Service (Amazon S3). Em seguida, crie um aplicativo do Managed Service for Apache Flink. Você passa na localização do pacote de código, um fluxo de dados do Kinesis como fonte de dados de fluxo e, normalmente, um local de fluxo ou arquivo que recebe os dados processados do aplicativo.

Pré-requisitos para concluir os exercícios

Para concluir as etapas neste guia, você deve ter o seguinte:

- Java Development Kit (JDK) versão 8. Defina a variável do ambiente`JAVA_HOME` para apontar para o local de instalação do JDK.
- Recomendamos que você use um ambiente de desenvolvimento (como Eclipse Java Neon ou IntelliJ Idea) para desenvolver e compilar seu aplicativo.
- Cliente do Git. Instale o cliente do Git se você ainda não tiver feito isso.
- Apache Maven Compiler Plugin. Maven deve estar em seu caminho de trabalho. Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

Para começar a usar, vá até Etapa 1: Configurar uma conta da AWS e criar um usuário administrador.

Etapa 1: Configurar uma conta da AWS e criar um usuário administrador

Cadastrar-se em uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas a seguir para criar uma.

Para se cadastrar em uma Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

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

   Quando você se cadastra em uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua acesso administrativo a um usuário administrativo e use somente o usuário raiz para realizar as tarefas que exigem acesso do usuário raiz.

A AWS envia um e-mail de confirmação depois que o processo de cadastramento é concluído. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário administrativo

Depois de se inscrever em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o AWS IAM Identity Center e crie um usuário administrativo para não usar o usuário raiz em tarefas cotidianas.

Proteger seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Console como o proprietário da conta ao escolher a opção Usuário raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, digite sua senha.

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

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

Criar um usuário administrativo

1. Habilitar o IAM Identity Center.

Para obter instruções, consulte Enabling AWS IAM Identity Center no Manual do Usuário do AWS IAM Identity Center.

2. No Centro de Identidade do IAM, conceda acesso administrativo a um usuário administrativo.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configure user access with the default Diretório do Centro de Identidade do IAM no Manual do Usuário do AWS IAM Identity Center.

Login como usuário administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use a URL de login que foi enviada ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda com o login utilizando um usuário do Centro de Identidade do IAM, consulte Fazer login no portal de acesso da AWS, no Guia do usuário do Início de Sessão da AWS.

Conceder acesso programático

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

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

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Configuração da AWS CLI para usar o AWS IAM Identity Center no AWS Command Line Interface Guia do usuário da .

Qual usuário precisa de acesso programático?	Para	Por
		• Para os SDKs da AWS, ferramentas e APIs da AWS, consulte Autenticação do Centro de Identidade do IAM no Guia de referência de ferramentas e SDKs da AWS.
IAM	Use credenciais temporárias para assinar solicitações programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções em Como usar credenciais temporárias com recursos da AWS no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para a AWS CLI, os SDKs da AWS ou as APIs da AWS.	Siga as instruções da interface que deseja utilizar. • Para a AWS CLI, consulte Autenticação usando as credenciais de usuário do IAM no Guia do usuário da AWS Command Line Interface. • Para as ferramentas e SDKs da AWS, consulte Autenticação usando as credenciais de longo prazo no Guia de referência de ferramentas e SDKs da AWS. • Para as APIs da AWS, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do IAM.

Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura a AWS CLI para uso com o Managed Service for Apache Flink.

> **Note**
>
> Os exercícios de conceitos básicos neste guia pressupõem que você esteja usando credenciais de administrador (`adminuser`) em sua conta para executar as operações.

> **Note**
>
> Se você já tiver a AWS CLI instalada, pode ser necessário atualizá-la para obter as funcionalidades mais recentes. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface. Para verificar a versão da AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios neste tutorial requerem a seguinte versão da AWS CLI ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar a AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Guia do usuário do AWS Command Line Interface:
   - Instalar a AWS Command Line Interface
   - Configurar a AWS CLI
2. Adicione um perfil nomeado para o usuário administrador no arquivo AWS CLI `config`. Você pode usar esse perfil ao executar os comandos da AWS CLI. Para obter mais informações sobre perfis nomeados, consulte Perfis nomeados no Guia do usuário da 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 obter uma lista das regiões da AWS disponíveis, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

> **Note**
>
> O código e os comandos de exemplo neste tutorial usam a região Oeste dos EUA (Oregon). Para usar uma região diferente, altere a região no código e nos comandos deste tutorial para a região que você deseja usar.

3. Verifique a configuração digitando o seguinte comando no prompt de comando:

```
aws help
```

Depois de configurar uma AWS conta e aAWS CLI, você pode tentar o próximo exercício, no qual você configura um aplicativo de amostra e testa a end-to-end configuração.

Próxima etapa

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Etapa 3: criar e executar um aplicativo Managed Service for Apache Flink

Neste exercício, você cria um aplicativo Managed Service for Apache Flink com fluxos de dados como origem e coletor.

Esta seção contém as seguintes etapas:

- Criar dois fluxos de dados do Amazon Kinesis Data Streams
- Gravação de registros de amostra no fluxo de entrada
- Baixar e examinar o código Java Apache Flink Streaming
- Compilar o código do aplicativo
- faça o upload do código Java Apache Flink Streaming
- Criar e executar o aplicativo do Managed Service for Apache Flink

Criar dois fluxos de dados do Amazon Kinesis Data Streams

Antes de criar um aplicativo do Managed Service for Apache Flink para este exercício, crie dois fluxos de dados do Kinesis (`ExampleInputStream` e `ExampleOutputStream`). O aplicativo usa esses fluxos para os fluxos de origem e de destino do aplicativo.

Você pode criar esses fluxos usando o console do Amazon Kinesis ou o comando da AWS CLI a seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro fluxo (`ExampleInputStream`), use o comando `create-stream` AWS CLI do Amazon Kinesis a seguir.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Para criar o segundo fluxo que o aplicativo usa para gravar a saída, execute o mesmo comando, alterando o nome da transmissão para `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Gravação de registros de amostra no fluxo de entrada

Nesta seção, você usa um script Python para gravar registros de amostra no fluxo para o aplicativo processar.

> **Note**
>
> Essa seção requer AWS SDK for Python (Boto).

1. Crie um arquivo denominado `stock.py` com o conteúdo a seguir:

```
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. Mais adiante neste tutorial, você executa o script `stock.py` para enviar dados para o aplicativo.

```
$ python stock.py
```

Baixar e examinar o código Java Apache Flink Streaming

O código do aplicativo Java para este exemplo está disponível em GitHub. Para fazer download do código do aplicativo, faça o seguinte:

1. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navegue até o diretório `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6`.

Observe o seguinte sobre o código do aplicativo:

- Um arquivo Project Object Model (pom.xml) contém informações sobre a configuração e as dependências do aplicativo, incluindo as bibliotecas do Managed Service for Apache Flink.
- O arquivo `BasicStreamingJob.java` contém o método `main` que define a funcionalidade do aplicativo.
- O aplicativo usa uma origem do Kinesis para ler o fluxo de origem. O trecho a seguir cria a origem do Kinesis:

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

- Seu aplicativo cria conectores de origem e de destino para acessar recursos externos usando um objeto `StreamExecutionEnvironment`.
- O aplicativo cria conectores de origem e de destino usando propriedades estáticas. Para usar as propriedades do aplicativo dinâmico, use os métodos `createSinkFromApplicationProperties` e `createSourceFromApplicationProperties` para criar os conectores. Esses métodos leem as propriedades do aplicativo para configurar os conectores.

  Para obter mais informações sobre as propriedades de runtime, consulte Propriedades de runtime.

Compilar o código do aplicativo

Nesta seção, você usa o compilador do Apache Maven para criar o código Java para o aplicativo. Para obter informações sobre como instalar o Apache Maven e o Java Development Kit (JDK), consulte Pré-requisitos para concluir os exercícios.

> **Note**
>
> Para usar o conector do Kinesis com versões do Apache Flink anteriores a 1.11, você precisa baixar o código-fonte do conector e compilá-lo conforme descrito na documentação do Apache Flink.

Para compilar o código do aplicativo

1. Para usar o seu código de aplicativo, compile-o e empacote-o em um arquivo JAR. Há duas formas de compilar e empacotar o código:

   - Use a ferramenta de linha de comando do Maven. Crie seu arquivo JAR executando o seguinte comando no diretório que contém o arquivo `pom.xml`:

     ```
     mvn package
     ```

     > **Note**
     >
     > O parâmetro -Dflink.version não é necessário para o runtime do Managed Service for Apache Flink versão 1.0.1; ele só é necessário para a versão 1.1.0 e posterior. Para ter mais informações, consulte the section called “Especificar a versão do Apache Flink do seu aplicativo”.

   - Use o ambiente de desenvolvimento. Consulte a documentação de seu ambiente de desenvolvimento para obter mais detalhes.

   Você pode carregar o pacote como um arquivo JAR, ou pode compactar o pacote e carregá-lo como um arquivo ZIP. Se você criar seu aplicativo usando a AWS CLI, especifique o tipo de conteúdo de código (JAR ou ZIP).
2. Se houver erros durante a compilação, verifique se sua variável de ambiente `JAVA_HOME` está definida corretamente.

Se o aplicativo for compilado com êxito, o arquivo a seguir é criado:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

faça o upload do código Java Apache Flink Streaming

Nesta seção, você cria um bucket do Amazon Simple Storage Service (Amazon S3) e faz upload do código do aplicativo.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.

2. Selecione Criar bucket.
3. Insira **ka-app-code-*<username>*** no campo Nome do bucket. Adicione um sufixo para o nome do bucket, como o nome do usuário, para torná-lo globalmente exclusivo. Selecione Next (Próximo).
4. Na etapa Configure options (Configurar opções), mantenha as configurações como estão e selecione Next (Próximo).
5. Na etapa Set permissions (Definir permissões), mantenha as configurações como estão e selecione Next (Próximo).
6. Selecione Criar bucket.
7. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Select files (Selecionar arquivos), selecione Add files (Adicionar arquivos). Navegue até o arquivo `aws-kinesis-analytics-java-apps-1.0.jar` que você criou na etapa anterior. Selecione Next (Próximo).
9. Na etapa Definir permissões, mantenha as configurações como estão. Escolha Próximo.
10. Na etapa Definir propriedades, mantenha as configurações como estão. Escolha Carregar.

O código passa a ser armazenado em um bucket do Amazon S3 que pode ser acessado pela aplicação.

Criar e executar o aplicativo do Managed Service for Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI.

> **Note**
>
> Quando você cria o aplicativo usando o console, seus recursos AWS Identity and Access Management (IAM) e do Amazon CloudWatch Logs são criados para você. Quando cria o aplicativo usando a AWS CLI, você cria esses recursos separadamente.

Tópicos

- Criar e executar o aplicativo (console)
- Criar e executar o aplicativo (AWS CLI)

Criar e executar o aplicativo (console)

Siga estas etapas para criar, configurar, atualizar e executar o aplicativo usando o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel do Managed Service for Apache Flink, selecione Criar aplicativo de análise.
3. Na página Managed Service for Apache Flink - Criar aplicativo, forneça os detalhes do aplicativo da seguinte forma:
   - Em Nome do aplicativo, insira **`MyApplication`**.
   - Em Descrição, insira **`My java test app`**.
   - Em Runtime, selecione Apache Flink.

     > **Note**
     >
     > O Managed Service for Apache Flink usa o Apache Flink versão 1.8.2 ou 1.6.2.

   - Altere o pulldown da versão para Apache Flink 1.6.
4. Em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um aplicativo Managed Service for Apache Flink usando o console, você tem a opção de ter um perfil do IAM e uma política criada para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses recursos do IAM são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Editar a política do IAM

Edite a política do IAM para adicionar permissões de acesso aos fluxos de dados do Kinesis.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-west-2`** que o console criou para você na seção anterior.
3. Na página Summary (Resumo), selecione Edit policy (Editar política). Selecione a guia JSON.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua os exemplos de IDs de conta (*012345678901*) pelo ID da conta.

```
{
    "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 o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na página Configurar aplicativo, forneça o Local do código:
   - Em Bucket do Amazon S3, insira **ka-app-code-*<username>***.
   - Em Caminho do objeto do Amazon S3, insira **java-getting-started-1.0.jar**.
3. Na seção Acesso aos recursos do aplicativo, em Permissões de acesso, selecione Criar/atualizar o perfil do IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Insira as seguintes propriedades e valores de aplicativo:

ID do grupo	Chave	Valor
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Em Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
6. Para CloudWatch registrar, marque a caixa de seleção Ativar.
7. Escolha Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro na Amazon, o Managed Service for Apache Flink cria um grupo de logs e um stream de logs para você. Os nomes desses recursos são os seguintes:
>
> - Grupo de logs: `/aws/kinesis-analytics/MyApplication`
> - Fluxo de logs: `kinesis-analytics-log-stream`

Executar o aplicativo

1. Na MyApplicationpágina, escolha Executar. Confirme a ação.
2. Quando o aplicativo estiver em execução, atualize a página. O console mostra o Gráfico do aplicativo.

Interromper o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualize o aplicativo

Usando o console, você pode atualizar configurações do aplicativo, como as propriedades do aplicativo, as configurações de monitoramento e a localização ou o nome do arquivo JAR do aplicativo. Você também pode recarregar o JAR do aplicativo do bucket do Amazon S3 se precisar atualizar o código do aplicativo.

Na MyApplicationpágina, escolha Configurar. Atualize as configurações do aplicativo e selecione Update (Atualizar).

Criar e executar o aplicativo (AWS CLI)

Nesta seção, você usa a AWS CLI para criar e executar o aplicativo Flink do Managed Service for Apache Flink. O Managed Service for Apache Flink usa o comando `kinesisanalyticsv2` AWS CLI para criar e interagir com aplicativos Managed Service for Apache Flink.

Criar uma política de permissões

Primeiro, crie uma política de permissões com duas instruções: uma que concede permissões para a ação `read` no fluxo de origem, e outra que concede permissões para ações `write` no fluxo de destino. Em seguida, você anexa a política a um perfil do IAM (que criará na próxima seção). Assim, ao assumir o perfil, o serviço Managed Service for Apache Flink terá as permissões necessárias para ler o fluxo de origem e gravar no fluxo de coleta.

Use o código a seguir para criar a política de permissões `AKReadSourceStreamWriteSinkStream`. Substitua *`username`* pelo nome de usuário que você usou para criar o bucket do Amazon S3 e armazenar o código do aplicativo. Substitua o ID da conta nos Nomes de recurso da Amazon (ARNs) (*`012345678901`*) pelo ID da conta.

```
{
    "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 obter instruções sobre como criar uma política de permissões, consulte Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia do usuário do IAM.

> **Note**
>
> Para acessar outros serviços da Amazon, você pode usar o AWS SDK for Java. O Managed Service for Apache Flink define automaticamente as credenciais exigidas pelo SDK como as credenciais do perfil do IAM associado a seu aplicativo. Não é necessária nenhuma etapa adicional.

Criar uma perfil do IAM

Nesta seção, você cria um perfil do IAM que o aplicativo Managed Service for Apache Flink pode assumir para ler um fluxo de origem e gravar no fluxo de coleta.

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Essas permissões são concedidas usando um perfil do IAM. Cada perfil do IAM tem duas políticas anexadas. A política de confiança concede ao Managed Service for Apache Flink permissão para assumir o perfil, e a política de permissões determina o que o serviço pode fazer depois de assumir a função.

Você anexa a política de permissões que criou na seção anterior a essa função.

Para criar uma perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Roles (Funções) e Create Role (Criar função).
3. Em Selecionar tipo de identidade de confiança, selecione Serviço da AWS. Em Choose the service that will use this role (Selecionar o serviço que usará esta função), selecione Kinesis. Em Select your use case (Selecionar seu caso de uso), selecione Kinesis Analytics.

   Selecione Next: Permissions (Próximo: permissões).
4. Na página Attach permissions policies, selecione Next: Review. Você pode anexar políticas de permissões depois de criar a função.
5. Na página Create role (Criar função), insira **`MF-stream-rw-role`** para o Role name (Nome da função). Selecione Criar função.

   Você criou um perfil do IAM chamado `MF-stream-rw-role`. Em seguida, você atualiza as políticas de confiança e de permissões para a função.
6. Anexe a política de permissões à função.

   > **Note**
   >
   > Para este exercício, o Managed Service for Apache Flink assume esse perfil para ler dados de um fluxo de dados do Kinesis (origem) e gravar a saída em outro fluxo de dados do Kinesis. Depois, você anexa a política que criou na etapa anterior, the section called “Criar uma política de permissões”.

   a. Na página Summary (Resumo), selecione a guia Permissions (Permissões).

   b. Selecione Attach Policies.

   c. Na caixa de pesquisa, insira **`AKReadSourceStreamWriteSinkStream`** (a política que você criou na seção anterior).

   d. Escolha a ReadSourceStreamWriteSinkStream política AK e escolha Anexar política.

Agora você criou a função de execução de serviço que seu aplicativo usa para acessar os recursos. Anote o ARN da nova função.

Para step-by-step obter instruções sobre como criar uma função, consulte Como criar uma função do IAM (console) no Guia do usuário do IAM.

Criar o aplicativo do Managed Service for Apache Flink

1. Salve o seguinte código JSON em um arquivo chamado `create_request.json`. Substitua o ARN da função de amostra pelo ARN da função que você criou anteriormente. Substitua o sufixo do ARN do bucket (*username*) pelo sufixo que você selecionou na seção anterior. Substitua o ID da conta de exemplo (*012345678901*) na função de execução do serviço pelo ID da conta.

```
{
    "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. Execute a ação `CreateApplication` com a solicitação anterior para criar o aplicativo:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

O aplicativo agora é criado. Você inicia o aplicativo na próxima etapa.

Iniciar o aplicativo

Nesta seção, você usa a ação `StartApplication` para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `start_request.json`.

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

2. Execute a ação `StartApplication` com a solicitação anterior para iniciar o aplicativo:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

O aplicativo agora está em execução. Você pode verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Interromper o aplicativo

Nesta seção, você usa a ação `StopApplication` para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Execute a ação `StopApplication` com a seguinte solicitação para interromper o aplicativo:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

O aplicativo agora está interrompido.

Adicionar uma opção de CloudWatch registro

Você pode usar o AWS CLI para adicionar um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter informações sobre como usar o CloudWatch Logs com seu aplicativo, consultethe section called “Configurando o log”.

Atualizar propriedades do ambiente

Nesta seção, você usa a ação `UpdateApplication` para alterar as propriedades do ambiente do aplicativo sem recompilar o código do aplicativo. Neste exemplo, você altera a região dos fluxos de origem e destino.

Para atualizar propriedades de ambiente para o aplicativo

1. Salve o seguinte código JSON em um arquivo chamado `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. Execute a ação `UpdateApplication` com a solicitação anterior para atualizar as propriedades do ambiente:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote do código, use a ação `UpdateApplication`AWS CLI.

Para usar o AWS CLI, exclua seu pacote do código anterior do bucket do Amazon S3, faça o upload da nova versão e chame `UpdateApplication`, especificando o mesmo nome de objeto e bucket do Amazon S3. O aplicativo será reiniciado com o novo pacote de código.

O exemplo de solicitação da `UpdateApplication` ação a seguir recarrega o código do aplicativo e reinicia o aplicativo. Atualize o `CurrentApplicationVersionId` para a versão atual do aplicativo. Você pode verificar a versão atual do aplicativo usando as ações `DescribeApplication` ou `ListApplications`. Atualize o sufixo do nome do bucket (*<username>*) com o sufixo que você selecionou na seção the section called “Criar dois fluxos de dados do 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"
                }
            }
        }
    }
}
```

Etapa 4: limpar os recursos AWS

Esta seção inclui procedimentos para limpar os recursos AWS criados no tutorial Introdução.

Este tópico contém as seguintes seções:

- Excluir o seu aplicativo Managed Service for Apache Flink
- Excluir seus fluxos de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Excluir seus recursos do IAM
- Exclua seus CloudWatch recursos

Excluir o seu aplicativo Managed Service for Apache Flink

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Managed Service for Apache Flink, escolha. MyApplication
3. Selecione Configurar.
4. Na seção Snapshots, selecione Desativar e, em seguida, selecione Atualizar.
5. Na página do aplicativo, selecione Excluir e, em seguida, confirme a exclusão.

Excluir seus fluxos de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
2. No painel Kinesis Data Streams, escolha. ExampleInputStream
3. Na ExampleInputStreampágina, escolha Excluir Kinesis Stream e confirme a exclusão.
4. Na página Kinesis Streams, escolha o, escolha Ações ExampleOutputStream, escolha Excluir e confirme a exclusão.

Excluir seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha o balde ka-app-code -. <username>
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Excluir seus recursos do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. Na barra de navegação, selecione Policies.
3. No controle do filtro, insira kinesis.
4. Escolha a política kinesis-analytics-service- MyApplication -us-west-2.
5. Selecione Ações da política e, em seguida, Excluir.
6. Na barra de navegação, selecione Roles (Funções).
7. Escolha a função kinesis-analytics- MyApplication -us-west-2.
8. Selecione Excluir função e, em seguida, confirme a exclusão.

Exclua seus CloudWatch recursos

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Logs.
3. Escolha o grupo de registros MyApplication/aws/kinesis-analytics/.
4. Selecione Excluir grupo de logs e confirme a exclusão.

Configurações do Apache Flink

O Managed Service for Apache Flink é uma implementação da estrutura Apache Flink. O Managed Service for Apache Flink usa os valores padrão descritos nesta seção. Alguns desses valores podem ser definidos pelos aplicativos Managed Service for Apache Flink em código, e outros não podem ser alterados.

Este tópico contém as seguintes seções:

Configuração do Apache Flink

O Managed Service for Apache Flink fornece uma configuração padrão do Flink que consiste em valores recomendados pelo Apache Flink para a maioria das propriedades e alguns baseados em perfis comuns de aplicativos. Para obter mais informações sobre a configuração do Flink, consulte Configuração. A configuração padrão fornecida pelo serviço funciona para a maioria dos aplicativos. No entanto, se precisar ajustar as propriedades de configuração do Flink para melhorar o desempenho de determinados aplicativos com alto paralelismo, alto uso de memória e estado, ou habilitar novos recursos de depuração no Apache Flink, você poderá alterar determinadas propriedades solicitando um caso de suporte. Para obter mais informações, consulte o Support Center do AWS. Você pode verificar a configuração atual do seu aplicativo usando o painel do Apache Flink.

Estado de back-end

O Managed Service for Apache Flink armazena dados transitórios em um estado de back-end. O Managed Service for Apache Flink usa o RocksDBStateBackend. Chamar `setStateBackend` para definir um back-end diferente não tem efeito.

Habilitamos os seguintes recursos no estado de back-end:

- Snapshots incrementais de estado de back-end
- Snapshots de estado assíncrono de back-end
- Recuperação local dos pontos de verificação

No Managed Service for Apache Flink, a configuração `state.backend.rocksdb.ttl.compaction.filter.enabled` é ativada por padrão. Usando esse filtro, você pode atualizar o código do aplicativo para ativar a estratégia de limpeza da compactação. Para obter mais informações, consulte Estado TTL no Flink 1.8.0 na documentação do Apache Flink.

Para obter mais informações sobre estados de back-end, consulte State Backends na documentação do Apache Flink.

Pontos de verificação

O Managed Service for Apache Flink usa uma configuração de ponto de verificação padrão com os seguintes valores. Alguns desses valores podem ser alterados. Você deve definir CheckpointConfiguration.ConfigurationType para `CUSTOM` para que o Managed Service for Apache Flink use valores de pontos de verificação modificados.

Configuração	Pode ser modificada?	Como	Valor padrão
CheckpointingEnabled	Permite modificação	Criar aplicativo Atualizar um aplicativo AWS CloudFormation	Verdadeiro
CheckpointInterval	Permite modificação	Criar aplicativo	60000

Configuração	Pode ser modificada?	Como	Valor padrão
		Atualizar um aplicativo AWS CloudFormation	
MinPauseBetweenCheckpoints	Permite modificação	Criar aplicativo Atualizar um aplicativo AWS CloudFormation	5000
Pontos de verificação não alinhados	Permite modificação	Caso de suporte	Falso
Número de pontos de verificação simultâneos	Não modificável	N/D	1
Modo de verificação	Não modificável	N/D	Exatamente uma vez
Política de retenção do ponto de verificação	Não modificável	N/D	Em caso de falha
Tempo limite do ponto de verificação	Não modificável	N/D	60 minutos
Número máximo de pontos de verificação retidos	Não modificável	N/D	1
Estratégia de reinicialização	Não modificável	N/D	Atraso fixo, com repetições infinitas a cada 10 segundos.

Configuração	Pode ser modificada?	Como	Valor padrão
Localização do ponto de verificação e do ponto de salvamento	Não modificável	N/D	Armazenamos dados duráveis de pontos de verificação e pontos de salvament o em um bucket S3 de propriedade do serviço.
Limite de memória do estado de back-end	Não modificável	N/D	1048576

Salvamento

Por padrão, ao restaurar a partir de um ponto de salvamento, a operação de retomada tentará mapear todo o estado do ponto de salvamento de volta até o programa com o qual você está restaurando. Se você descartar um operador, por padrão, a restauração a partir de um ponto de salvamento que tenha dados que correspondam ao operador ausente falhará. Você pode permitir que a operação seja bem-sucedida definindo o parâmetro allowNonRestoredState da FlinkRunConfiguration do aplicativo como `true`. Isso permitirá que a operação de retomada ignore um estado que não possa ser mapeado para o novo programa.

Para obter mais informações, consulte Allowing Non-Restored State (Permitir estado não restaurado) na documentação do Apache Flink.

Tamanhos de heap

O Managed Service for Apache Flink aloca cada KPU 3 GiB do heap da JVM e reserva 1 GiB para alocações de código nativo. Para obter informações sobre como aumentar a capacidade do seu aplicativo, consulte the section called “Escalabilidade”.

Para obter mais informações sobre tamanhos de heap da JVM, consulte Configuração na documentação do Apache Flink.

Diminuição do buffer

A diminuição do buffer pode ajudar aplicativos com alta contrapressão. Se o seu aplicativo apresentar falhas nos pontos de verificação/salvamento, habilitar esse recurso pode ser útil. Para fazer isso, solicite um caso de suporte.

Para obter mais informações, consulte The Buffer Debloating Mechanism na documentação do Apache Flink.

Propriedades de configuração modificáveis do Flink

A seguir estão as configurações do Flink que você pode modificar usando um caso de suporte. Você pode modificar mais de uma propriedade por vez e para vários aplicativos ao mesmo tempo especificando o prefixo do aplicativo. Se houver outras propriedades de configuração do Flink fora dessa lista que você queira modificar, especifique a propriedade exata no seu caso.

Tolerância a falhas

`restart-strategy:`

`restart-strategy.fixed-delay.delay:`

Pontos de verificação e estados de back-end

`state.backend:`

`state.backend.fs.memory-threshold:`

`state.backend.incremental:`

Pontos de verificação

`execution.checkpointing.unaligned:`

Métricas nativas do RocksDB

As métricas nativas do RocksDB não são enviadas para CloudWatch. Depois de ativadas, essas métricas podem ser acessadas no painel do Flink ou na API REST do Flink com ferramentas personalizadas.

O Managed Service for Apache Flink permite que os clientes acessem a API REST mais recente do Flink (ou a versão compatível que você está usando) no modo de somente leitura usando a API CreateApplicationPreSignedURL. Essa API é usada pelo próprio painel do Flink, mas também pode ser usada por ferramentas de monitoramento 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:
```

Opções avançadas de estados de back-end

```
state.storage.fs.memory-threshold:
```

Opções completas do 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:

Configuração de memória

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:

Opções avançadas de cluster

cluster.intercept-user-system-exit:

cluster.processes.halt-on-fatal-error:

Configurações do Filesystem

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

Opções avançadas de tolerância a falhas

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

Configuração de memória

`jobmanager.memory.heap.size:`

Métricas

`metrics.latency.interval:`

Opções avançadas para o endpoint REST e cliente

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Opções avançadas de segurança SSL

`security.ssl.internal.handshake-timeout:`

Opções avançadas de programação

`slot.request.timeout:`

Opções avançadas para interface do usuário da web do Flink

`web.timeout:`

Visualizar propriedades configuradas do Flink

Você pode visualizar as propriedades do Apache Flink que você mesmo configurou ou solicitou que fossem modificadas por meio de um caso de suporte por meio do painel do Apache Flink e seguindo estas etapas:

1. Acesse o painel do Flink
2. Selecione Job Manager no painel de navegação do lado esquerdo.
3. Selecione Configuração para ver a lista de propriedades do Flink.

Configurar o Managed Service for Apache Flink para acessar recursos em uma Amazon VPC

É possível configurar um aplicativo Managed Service for Apache Flink para se conectar a sub-redes privadas em uma nuvem privada virtual (VPC) em sua conta. Use a Amazon Virtual Private Cloud (Amazon VPC) para criar uma rede privada para recursos, como bancos de dados, instâncias de cache ou serviços internos. Conecte seu aplicativo à VPC para acessar recursos privados durante a execução.

Este tópico contém as seguintes seções:

Conceitos da Amazon VPC

A Amazon VPC é a camada de rede do Amazon EC2. Se você não tem experiência com o Amazon EC2, consulte O que é o Amazon EC2? no Manual do usuário do Amazon EC2 para instâncias do Linux para obter uma breve visão geral.

Veja a seguir os principais conceitos das VPCs:

- Uma nuvem privada virtual (VPC) é uma rede virtual dedicada à sua conta da AWS.
- Uma sub-rede é uma gama de endereços IP na VPC.
- Uma tabela de rotas contém um conjunto de regras, chamado de rotas, que são usadas para determinar para onde o tráfego de rede é direcionado.
- Um gateway da Internet é um componente da VPC horizontalmente dimensionado, redundante e altamente disponível que permite a comunicação entre instâncias na VPC e a Internet. Portanto, não impõe nenhum risco de disponibilidade ou restrições de largura de banda no tráfego da rede.
- Um endpoint da VPC permite conectar de forma privada a VPC aos serviços compatíveis da AWS e aos serviços do endpoint da VPC desenvolvidos pelo PrivateLink sem exigir um gateway da

Internet, um dispositivo NAT, uma conexão VPN ou uma conexão do AWS Direct Connect. As instâncias na sua VPC não exigem que endereços IP públicos se comuniquem com recursos no serviço. O tráfego entre a sua VPC e os outros serviços não deixa a rede da Amazon.

Para receber mais informações sobre o serviço Amazon VPC, consulte o Guia do usuário do Amazon Virtual Private Cloud.

O Managed Service for Apache Flink cria interfaces de rede elásticas em uma das sub-redes fornecidas em sua configuração de VPC para o aplicativo. O número de interfaces de rede elástica criadas em suas sub-redes VPC pode variar, dependendo do paralelismo e do paralelismo por KPU do aplicativo. Para obter mais informações sobre a escalabilidade do aplicativo, consulte Escalabilidade.

> **Note**
>
> As configurações de VPC não são compatíveis com aplicativos SQL.

> **Note**
>
> O serviço Managed Service for Apache Flink gerencia o estado do ponto de verificação e do instantâneo para aplicativos que têm uma configuração de VPC.

Permissões para aplicativo VPC

Esta seção descreve as políticas de permissão que seu aplicativo precisará para funcionar com sua VPC. Para obter informações sobre o uso de políticas de permissões, consulte Gerenciamento de identidade e acesso para o Amazon Managed Service for Apache Flink.

A política de permissões a seguir concede ao seu aplicativo as permissões necessárias para interagir com uma VPC. Para usar essa política de permissão, adicione-a à função de execução do seu aplicativo.

Política de permissões para acessar uma 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**
>
> Quando você especifica recursos do aplicativo usando o console (como o CloudWatch Logs ou um Amazon VPC), o console modifica sua função de execução do aplicativo para conceder permissão para acessar esses recursos. Você só precisa modificar manualmente a função de execução do aplicativo se criar o aplicativo sem usar o console.

Acesso à Internet e serviços para um aplicativo Managed Service for Apache Flink conectado à VPC

Por padrão, quando você conecta um aplicativo Managed Service for Apache Flink a uma VPC em sua conta, ele não terá acesso à internet, a menos que a VPC forneça acesso. Se o aplicativo precisar de acesso à internet, o seguinte deverá ser verdadeiro:

- O aplicativo Managed Service for Apache Flink só deve ser configurado com sub-redes privadas.
- A VPC deve conter um gateway ou instância NAT em uma sub-rede pública.
- Deve existir uma rota para tráfego de saída das sub-redes privadas para o gateway NAT em uma sub-rede pública.

> **Note**
>
> Vários serviços oferecem VPC endpoints. É possível usar endpoints da VPC para se conectar aos serviços da Amazon de dentro de uma VPC sem acesso à internet.

Se uma sub-rede é pública ou privada depende de sua tabela de rotas. Cada tabela de rotas tem uma rota padrão, que determina o próximo salto para pacotes que têm um destino público.

- Para uma sub-rede privada: a rota padrão aponta para um gateway NAT (nat-...) ou instância NAT (eni-...).
- Para uma sub-rede pública: a rota padrão aponta para um gateway da internet (igw-...).

Depois de configurar sua VPC com uma sub-rede pública (com NAT) e uma ou mais sub-redes privadas, faça o seguinte para identificar suas sub-redes públicas e privadas:

- No painel de navegação do console da VPC, selecione Sub-redes.
- Selecione uma sub-rede e depois a guia Tabela de rotas. Verifique a rota padrão:
  - Sub-rede pública: Destino: 0.0.0.0/0, Destino: igw-...
  - Sub-rede privada: Destino: 0.0.0.0/0, Alvo: nat-... ou eni-...

Para associar o aplicativo Managed Service for Apache Flink a sub-redes privadas:

- Abra o console do Managed Service for Apache Flink em https://console.aws.amazon.com/flink
- Na página dos Aplicativos do Managed Service for Apache Flink, selecione seu aplicativo e depois Detalhes do aplicativo.
- Na página do seu aplicativo, selecione Configurar.
- Na seção Conectividade da VPC, selecione a VPC a ser associada ao aplicativo. Selecione as sub-redes e o grupo de segurança associados à sua VPC os quais você deseja que o aplicativo use para acessar os recursos da VPC.
- Escolha Atualizar.

Informações relacionadas

Criar uma VPC com sub-redes públicas e privadas

Conceitos básicas do gateway NAT

API de VPC do Managed Service for Apache Flink

Use as seguintes operações de API do Managed Service for Apache Flink para gerenciar VPCs para seu aplicativo. Para obter informações sobre como usar a API do Managed Service for Apache Flink, consulte Exemplos de códigos de API.

CreateApplication

Use a ação CreateApplication para adicionar uma configuração de VPC ao seu aplicativo durante a criação.

O exemplo a seguir de código de solicitação para a ação `CreateApplication` inclui uma configuração de VPC quando o aplicativo é criado:

```
{
  "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

Use a ação CreateApplication para adicionar uma configuração de VPC ao seu aplicativo durante a criação.

O exemplo a seguir de código de solicitação para a ação `AddApplicationVpcConfiguration` adiciona uma configuração de VPC a um aplicativo existente:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

DeleteApplicationVpcConfiguration

Use a ação DeleteApplicationVPCConfiguration para remover uma configuração de VPC do seu aplicativo.

O exemplo de solicitação a seguir para a ação `AddApplicationVpcConfiguration` remove tags de um aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

UpdateApplication

Use a ação UpdateApplication para atualizar todas as configurações de VPC de um aplicativo de uma só vez.

O exemplo de código de solicitação a seguir para a ação `UpdateApplication` atualiza todas as configurações de VPC de um aplicativo:

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

Exemplo: uso de uma VPC para acessar dados em um cluster do Amazon MSK

Para obter um tutorial completo sobre como acessar dados de um cluster do Amazon MSK em uma VPC, consulte Replicação do MSK.

Solução de problemas do Managed Service for Apache Flink

As considerações a seguir podem ajudar a solucionar problemas que possam surgir com o Amazon Managed Service for Apache Flink.

Tópicos

Solução de problemas do desenvolvimento

Tópicos

Gráficos de chama do Apache Flink

Os gráficos de chama são habilitados por padrão em aplicativos nas versões do Managed Service for Apache Flink compatíveis com ele. Os gráficos de chama podem afetar o desempenho do aplicativo se você mantiver o gráfico aberto, conforme mencionado na documentação do Flink.

Se você quiser desativar o gráficos de chama para seu aplicativo, crie um caso para solicitar que ele seja desativado para o ARN do seu aplicativo. Para obter mais informações, consulte o AWS Support Center.

Problema do provedor de credenciais com o conector EFO 1.15.2

Há um problema conhecido com as versões do conector EFO do Kinesis Data Streams até 1.15.2 em que `FlinkKinesisConsumer` não está respeitando a configuração `Credential Provider`. As configurações válidas estão sendo desconsideradas devido ao problema, o que resulta no uso do provedor de credenciais AUTO. Isso pode causar um problema ao usar o acesso entre contas para o Kinesis usando o conector EFO.

Para resolver esse erro, use o conector EFO versão 1.15.3 ou superior.

Aplicativos com conectores Kinesis não compatíveis

O Managed Service for Apache Flink versão 1.15 rejeitará automaticamente a inicialização ou atualização dos aplicativos se eles estiverem usando versões incompatíveis do Kinesis Connector (versão anterior a 1.15.2) agrupadas em JARs ou arquivos de aplicativos (ZIP).

Erro de rejeição

Você verá o seguinte erro ao enviar chamadas de criação e atualização do aplicativo por meio de:

```
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
```

Etapas para remediar

- Atualize a dependência do aplicativo em `flink-connector-kinesis`. Se você estiver usando o Maven como ferramenta de construção do seu projeto, siga Atualizar uma dependência do Maven . Se você estiver usando o Gradle, siga Atualizar uma dependência do Gradle .
- Reempacote o aplicativo.
- Faça upload para um bucket do Amazon S3.
- Reenvie a solicitação de criação/atualização do aplicativo com o aplicativo revisado que acabou de ser carregado no bucket do Amazon S3.
- Se você continuar vendo a mesma mensagem de erro, verifique novamente as dependências do aplicativo. Se o problema persistir, crie um ticket de suporte.

Atualizar uma dependência do Maven

1. Abra `pom.xml` do projeto.
2. Encontre as dependências do projeto. Elas se parecem com:

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

3. Atualize `flink-connector-kinesis` para uma versão igual ou posterior à 1.15.2. Por exemplo:

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.2</version>
        </dependency>
```

```
        ...

    </dependencies>

    ...

</project>
```

Atualizar uma dependência do Gradle

1. Abra `build.gradle` do projeto (ou `build.gradle.kts` para aplicativos Kotlin).
2. Encontre as dependências do projeto. Elas se parecem com:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. Atualize `flink-connector-kinesis` para uma versão igual ou posterior à 1.15.2. Por exemplo:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}
```

```
...
```

Erro de compilação: “Não foi possível resolver as dependências do projeto”

Para compilar os aplicativos de amostra do Managed Service for Apache Flink, você deve primeiro baixar e compilar o conector Apache Flink Kinesis e adicioná-lo ao seu repositório Maven local. Se o conector não tiver sido adicionado ao seu repositório, um erro de compilação semelhante ao seguinte será exibido:

```
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 esse erro, você deve baixar o código-fonte do Apache Flink (versão 1.8.2 em https://flink.apache.org/downloads.html) para o conector. Para obter instruções sobre como baixar, compilar e instalar o código-fonte do Apache Flink, consulte the section called “Usar o conector de fluxos Kinesis para o Apache Flink com versões anteriores do Apache Flink”.

Selecione inválida: “kinesisanalyticsv2"

Para usar a v2 da API do Managed Service for Apache Flink, você precisa da versão mais recente do AWS Command Line Interface (AWS CLI).

Para obter informações sobre como atualizar AWS CLI, consulte Instalação de AWS Command Line Interface no Guia do usuário de AWS Command Line Interface.

UpdateApplication A ação não está recarregando o código do aplicativo

A UpdateApplicationação não recarregará o código do aplicativo com o mesmo nome de arquivo se nenhuma versão do objeto S3 for especificada. Para recarregar o código do aplicativo com o mesmo nome de arquivo, habilite o versionamento em seu bucket do S3 e especifique a nova versão do objeto usando o parâmetro `ObjectVersionUpdate`. Para obter mais informações sobre habilitação de versionamento de objetos em um bucket do S3, consulte Habilitação e desativação de versionamento.

S3 StreamingFileSink FileNotFoundExceptions

O Managed Service for Apache Flink pode se deparar com `FileNotFoundException` em um arquivo parcial em andamento ao iniciar a partir de instantâneos se o arquivo parcial em andamento referido tiver seu ponto de salvamento ausente. Quando esse modo de falha ocorre, o estado do operador do aplicativo Managed Service for Apache Flink geralmente não é recuperável e deve ser reiniciado sem o instantâneo, usando `SKIP_RESTORE_FROM_SNAPSHOT`. Veja o seguinte exemplo 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
...
```

O Flink `StreamingFileSink` grava registros em sistemas de arquivos suportados pela abstração do Flink. FileSystem Como os fluxos de entrada podem não ser vinculados, os dados são organizados em arquivos parciais de tamanho finito com novos arquivos adicionados à medida que os dados são gravados. O ciclo de vida parcial e a política de sobreposição determinam o tempo, o tamanho e a nomenclatura dos arquivos parciais.

> **Note**
>
> Para obter mais informações, consulte Ciclo de vida de arquivos parciais.

Durante o ponto de verificação e o ponto de salvamento (captura instantânea), todos os arquivos pendentes são renomeados e confirmados. No entanto, os arquivos parciais em andamento não são confirmados, mas renomeados, e sua referência é mantida nos metadados do ponto de verificação ou do ponto de salvamento para serem usados na restauração de trabalhos. Esses arquivos parciais

em andamento acabarão sendo transferidos para pendentes, renomeados e confirmados por um ponto de verificação ou ponto de salvamento subsequente.

A seguir estão as principais causas e a mitigação da ausência do arquivo parcial em andamento:

- Snapshot obsoleto usado para iniciar o aplicativo Managed Service for Apache Flink — somente o último snapshot do sistema obtido quando um aplicativo é interrompido ou atualizado pode ser usado para iniciar um aplicativo Managed Service for Apache Flink com o Amazon S3. StreamingFileSink Para evitar essa classe de falha, use o instantâneo mais recente do sistema.
  - Isso acontece, por exemplo, quando você seleciona um instantâneo criado usando `CreateSnapshot`, em vez de um instantâneo acionado pelo sistema, durante a interrupção ou atualização. O ponto de salvamento do snapshot antigo mantém uma out-of-date referência ao arquivo de peça em andamento que foi renomeado e confirmado pelo ponto de verificação ou ponto de salvamento subsequente.
  - Isso também pode acontecer quando um instantâneo acionado pelo sistema de um evento de interrupção/atualização não mais recente é selecionado. Um exemplo é um aplicativo com o instantâneo do sistema desativado, mas `RESTORE_FROM_LATEST_SNAPSHOT` configurado. Geralmente, o Managed Service para aplicativos Apache Flink com o Amazon StreamingFileSink S3 deve sempre ter o snapshot do sistema ativado e configurado. `RESTORE_FROM_LATEST_SNAPSHOT`
- Arquivo parcial em andamento removido — Como o arquivo parcial em andamento está localizado em um bucket do S3, ele pode ser removido por outros componentes ou atores que tenham acesso ao bucket.
  - Isso pode acontecer quando você interrompe seu aplicativo por muito tempo e o arquivo de peça em andamento referido pelo ponto de salvamento do seu aplicativo foi removido pela política de ciclo de vida do bucket do S3. MultiPartUpload Para evitar essa classe de falha, certifique-se de que sua política de ciclo de vida do S3 Bucket MPU cubra um período suficientemente grande para seu caso de uso.
  - Isso também pode acontecer quando o arquivo parcial em andamento é removido manualmente ou por outro componente do sistema. Para evitar essa classe de falha, certifique-se de que os arquivos parciais em andamento não sejam removidos por outros atores ou componentes.
- Condição de corrida em que um ponto de verificação automatizado é acionado após o ponto de salvamento — Isso afeta as versões do Managed Service para Apache Flink até 1.13, inclusive. Esse problema foi corrigido no Managed Service for Apache Flink versão 1.15; migre seu aplicativo para o Managed Service for Apache Flink versão 1.15 para evitar a recorrência. Também sugerimos migrar de StreamingFileSink para FileSink.

- Quando os aplicativos são interrompidos ou atualizados, o Managed Service for Apache Flink aciona um ponto de salvamento e interrompe o aplicativo em duas etapas. Se um ponto de verificação automatizado for acionado entre as duas etapas, o ponto de salvamento ficará inutilizável, pois seu arquivo parcial em andamento será renomeado e potencialmente confirmado.

FlinkKafkaConsumer problema com stop with savepoint

Ao usar o legado, FlinkKafkaConsumer existe a possibilidade de seu aplicativo ficar preso em UPDATE, STOPPING ou SCALING, se você tiver os instantâneos do sistema habilitados. Não há nenhuma correção publicada disponível para esse problema, portanto, recomendamos que você atualize para a nova KafkaSourcepara mitigar esse problema.

Se você estiver usando `FlinkKafkaConsumer` com instantâneos habilitados, existe a possibilidade de, quando o trabalho do Flink processar uma solicitação da API de interrupção com ponto de salvamento, que `FlinkKafkaConsumer` possa falhar com um erro de runtime relatando `ClosedException`. Sob essas condições, o aplicativo Flink fica preso, manifestando-se como pontos de verificação com falha.

Deadlock do coletor assíncrono Flink 1.15

Há um problema conhecido com AWS conectores para a interface de implementação do Apache Flink. AsyncSink Isso afeta os aplicativos que usam o Flink 1.15 com os seguintes conectores:

- Para aplicativos 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`

- Aplicativos Flink SQL/TableAPI/Python:
  - 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`

Os aplicativos afetados apresentarão os seguintes sintomas:

- O trabalho do Flink está no estado de `RUNNING`, mas não está processando dados;
- Não há reinicializações do trabalho;
- Os pontos de controle estão atingindo o tempo limite.

O problema é causado por um bug no AWS SDK que faz com que ele não mostre certos erros ao chamador ao usar o cliente HTTP assíncrono. Isso faz com que o coletor espere indefinidamente pela conclusão de uma "solicitação em trânsito" durante uma operação de descarga no ponto de controle.

Esse problema foi corrigido no AWS SDK a partir da versão 2.20.144.

A seguir estão as instruções sobre como atualizar os conectores afetados para usar a nova versão do AWS SDK nos seus aplicativos:

Tópicos

- Atualizar aplicativos Java
- Atualizar aplicativos Python

Atualizar aplicativos Java

Siga os procedimentos abaixo para atualizar os aplicativos Java:

flink-connector-kinesis

Se o aplicativo usa `flink-connector-kinesis`:

O conector Kinesis usa sombreamento para empacotar algumas dependências, incluindo o AWS SDK, no jar de conectores. Para atualizar a versão do AWS SDK, use o procedimento a seguir para substituir essas classes sombreadas:

Maven

1. Adicione o conector Kinesis e os módulos AWS SDK necessários como dependências do projeto.
2. Configure `maven-shade-plugin`:

a. Adicione um filtro para excluir classes sombreadas de AWS SDK ao copiar o conteúdo do jar do conector Kinesis.

b. Adicione uma regra de realocação para mover as classes atualizadas do AWS SDK para o pacote, conforme esperado pelo conector 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. Adicione o conector Kinesis e os módulos AWS SDK necessários como dependências do projeto.
2. Ajuste a configuração do ShadowJar:

   a. Exclua classes de AWS SDK sombreadas ao copiar o conteúdo do jar do conector Kinesis.

   b. Realoque as classes AWS SDK atualizadas para um pacote esperado pelo conector 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")
}
...
```

Outros conectores afetados

Se o aplicativo usar outro conector afetado:

Para atualizar a versão do AWS SDK, a versão do SDK deve ser aplicada na configuração de compilação do projeto.

Maven

Adicione a lista de materiais (BOM) do AWS SDK à seção de gerenciamento de dependências do arquivo `pom.xml` para impor a versão do SDK para o projeto.

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

Adicione dependência de plataforma à lista de materiais (BOM) do AWS SDK para aplicar a versão do SDK para o projeto. Isso requer o Gradle 5.0 ou mais recente:

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

Atualizar aplicativos Python

Os aplicativos Python podem usar conectores de duas maneiras diferentes: empacotar conectores e outras dependências Java como parte de um único uber-jar ou usar o jar de conectores diretamente. Para corrigir aplicativos afetados pelo deadlock do Async Sink:

- Se o aplicativo usar um uber jar, siga as instruções para Atualizar aplicativos Java .
- Para recriar jars de conectores a partir da fonte, use as seguintes etapas:

Construir conectores a partir da fonte:

Pré-requisitos, semelhantes aos requisitos de compilação do Flink:

- Java 11
- Maven 3.2.5

flink-sql-connector-kinesis

1. Baixe o código-fonte do Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Descompactar código-fonte:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navegue até o diretório do conector kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. Compile e instale o jar de conectores, especificando a versão necessária do AWS SDK. Para acelerar a compilação, use `-DskipTests` para ignorar a execução do teste e `-Dfast` para ignorar verificações adicionais do código-fonte:

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. Navegue até o diretório do conector kinesis

```
cd ../flink-sql-connector-kinesis
```

6. Compile e instale o jar do conector sql:

```
mvn clean install -DskipTests -Dfast
```

7. O jar resultante estará disponível em:

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-fluxos de cinesia

1. Baixe o código-fonte do Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Descompactar código-fonte:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navegue até o diretório do conector kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. Compile e instale o jar de conectores, especificando a versão necessária do AWS SDK. Para acelerar a compilação, use `-DskipTests` para ignorar a execução do teste e `-Dfast` para ignorar verificações adicionais do código-fonte:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Navegue até o diretório do conector kinesis

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. Compile e instale o jar do conector sql:

```
mvn clean install -DskipTests -Dfast
```

7. O jar resultante estará disponível em:

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-quinesis-mangueira de incêndio

1. Baixe o código-fonte do Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Descompactar código-fonte:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navegue até o diretório do conector

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. Compile e instale o jar de conectores, especificando a versão necessária do AWS SDK. Para acelerar a compilação, use `-DskipTests` para ignorar a execução do teste e `-Dfast` para ignorar verificações adicionais do código-fonte:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Navegue até o diretório do conector sql

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. Compile e instale o jar do conector sql:

```
mvn clean install -DskipTests -Dfast
```

7. O jar resultante estará disponível em:

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. Baixe o código-fonte do 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. Descompactar código-fonte:

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. Navegue até o diretório do conector

```
cd flink-connector-aws-3.0.0
```

4. Compile e instale o jar de conectores, especificando a versão necessária do AWS SDK. Para acelerar a compilação, use `-DskipTests` para ignorar a execução do teste e `-Dfast` para ignorar verificações adicionais do código-fonte:

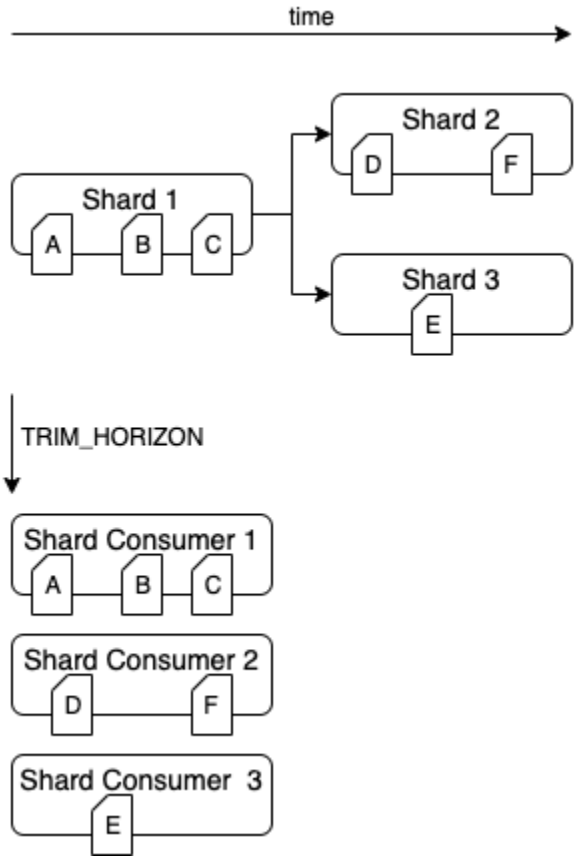
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -
Daws.sdk.version=2.20.144
```

5. O jar resultante estará disponível em:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

Processamento da fonte do Amazon Kinesis Data Streams fora de ordem durante a refragmentação

A FlinkKinesisConsumer implementação atual não oferece garantias sólidas de ordenação entre fragmentos do Kinesis. Isso pode levar ao out-of-order processamento durante a refragmentação do Kinesis Stream, especialmente para aplicativos Flink que apresentam atraso no processamento. Em algumas circunstâncias, por exemplo, os operadores de Windows baseados nos horários dos eventos, os eventos podem ser descartados devido ao atraso resultante.



Esse é um problema conhecido no Open Source Flink. Até que a correção do conector seja disponibilizada, verifique se os aplicativos Flink não estão mais lentos do que o Kinesis Data Streams durante o reparticionamento. Ao garantir que o atraso no processamento seja tolerado por seus aplicativos Flink, você pode minimizar o impacto do out-of-order processamento e o risco de perda de dados.

Solução de problemas de runtime

Esta seção contém informações sobre como diagnosticar e corrigir problemas de runtime com seu aplicativo Managed Service for Apache Flink.

Tópicos

Ferramentas de solução de problemas

A principal ferramenta para detectar problemas de aplicativos são os alarmes do CloudWatch. Usando os alarmes do CloudWatch, você pode definir limites para métricas do CloudWatch que indicam condições de erro ou gargalo em seu aplicativo. Para obter informações sobre os alarmes do CloudWatch recomendados, consulte Usando CloudWatch alarmes com o Amazon Managed Service para Apache Flink.

Problemas do aplicativo

Esta seção contém soluções para condições de erro que você pode encontrar com seu aplicativo Managed Service for Apache Flink.

Tópicos

O aplicativo está preso em um status transitório

Se seu aplicativo permanecer em um status transitório (`STARTING`, `UPDATING`, `STOPPING` ou `AUTOSCALING`), você poderá interrompê-lo usando a ação StopApplication com o parâmetro `Force` definido como `true`. Você não pode forçar a interrupção de um aplicativo no status `DELETING`. Como alternativa, se o aplicativo estiver no status `UPDATING` ou `AUTOSCALING`, você poderá revertê-lo para a versão anterior em execução. Quando você reverte um aplicativo, ele carrega dados do estado do último instantâneo bem-sucedido. Se o aplicativo não tiver instantâneos, o Managed Service for Apache Flink rejeitará a solicitação de reversão. Para obter mais informações sobre como reverter um aplicativo, consulte a ação RollbackApplication.

> **Note**
>
> A interrupção forçada do aplicativo pode levar à perda ou duplicação de dados. Para evitar a perda de dados ou o processamento duplicado de dados durante a reinicialização do aplicativo, recomendamos que você tire instantâneos frequentes do seu aplicativo.

As causas para aplicativos travados incluem o seguinte:

- O estado do aplicativo é muito grande: ter um estado de aplicativo muito grande ou muito persistente pode fazer com que o aplicativo fique preso durante uma operação de ponto de verificação ou instantâneo. Verifique as métricas `lastCheckpointDuration` e `lastCheckpointSize` do seu aplicativo para valores crescentes constantes ou valores anormalmente altos.
- O código do aplicativo é muito grande: verifique se o arquivo JAR do aplicativo tem menos de 512 MB. Arquivos JAR maiores que 512 MB não são suportados.
- Falha na criação do instantâneo do aplicativo: o Managed Service for Apache Flink tira um instantâneo do aplicativo durante uma solicitação de `UpdateApplication` ou `StopApplication`. Em seguida, o serviço usa esse estado de instantâneo e restaura o aplicativo usando a configuração atualizada do aplicativo para fornecer uma semântica de processamento exatamente uma vez. Se a criação automática do instantâneo falhar, consulte Falha na criação do instantâneo a seguir.
- Falha na restauração a partir de um instantâneo: se você remover ou alterar um operador em uma atualização do aplicativo e tentar restaurar a partir de um instantâneo, a restauração falhará por padrão se o instantâneo contiver dados de estado do operador ausente. Além disso, o aplicativo ficará preso no status `STOPPED` ou `UPDATING`. Para alterar esse comportamento e permitir que a restauração seja bem-sucedida, altere o parâmetro AllowNonRestoredState da FlinkRunConfiguration do aplicativo para `true`. Isso permitirá que a operação de retomada ignore dados de estado que não possam ser mapeados para o novo programa.
- A inicialização do aplicativo está demorando mais: o Managed Service for Apache Flink usa um tempo limite interno de 5 minutos (configuração suave) enquanto aguarda o início de uma tarefa do Flink. Se sua tarefa não começar dentro desse tempo limite, você verá um log do CloudWatch da seguinte forma:

  ```
  Flink job did not start within a total timeout of 5 minutes for application: %s under
   account: %s
  ```

  Se você encontrar o erro acima, isso significa que suas operações definidas no método `main` da tarefa do Flink estão demorando mais de 5 minutos, fazendo com que a criação da tarefa do Flink expire no final do Managed Service for Apache Flink. Sugerimos que você verifique os logs JobManager do Flink, bem como o código do seu aplicativo, para ver se esse atraso no método `main` é esperado. Caso contrário, você precisa adotar medidas para resolver o problema para que ele seja concluído em menos de 5 minutos.

Você pode verificar o status do seu aplicativo usando as ações `ListApplications` ou `DescribeApplication`.

Falha na criação do instantâneo

O serviço Managed Service for Apache Flink não pode tirar um instantâneo nas seguintes circunstâncias:

- O aplicativo excedeu o limite de instantâneos. O limite para instantâneos é de 1.000. Para obter mais informações, consulte Snapshots.
- O aplicativo não tem permissões para acessar sua fonte ou coletor.
- O código do aplicativo não está funcionando corretamente.
- O aplicativo está enfrentando outros problemas de configuração.

Se você receber uma exceção ao tirar um instantâneo durante uma atualização do aplicativo ou ao interromper o aplicativo, defina a propriedade `SnapshotsEnabled` de `ApplicationSnapshotConfiguration` do seu aplicativo como `false` e repita a solicitação.

Os instantâneos podem falhar se os operadores do seu aplicativo não forem provisionados adequadamente. Para obter informações sobre como ajustar o desempenho do operador, consulte Escalonamento operador.

Depois que o aplicativo retornar ao estado íntegro, recomendamos que você defina a propriedade `SnapshotsEnabled` do aplicativo como `true`.

Não é possível acessar recursos em uma VPC

Se seu aplicativo usa uma VPC em execução na Amazon VPC, faça o seguinte para verificar se seu aplicativo tem acesso aos recursos:

- Verifique se há o seguinte erro nos logs do CloudWatch. Esse erro indica que seu aplicativo não pode acessar recursos em sua VPC:

  ```
  org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after
   60000 ms.
  ```

  Se você ver esse erro, verifique se suas tabelas de rotas estão configuradas corretamente e se seus conectores têm as configurações de conexão corretas.

Para obter informações sobre a configuração e análise de logs do CloudWatch, consulte Registro e Monitoramento.

Dados são perdidos ao gravar em um bucket do Amazon S3

Pode ocorrer alguma perda de dados ao gravar a saída em um bucket do Amazon S3 usando o Apache Flink versão 1.6.2. Recomendamos usar a versão mais recente compatível do Apache Flink ao usar o Amazon S3 para saída direta. Para gravar em um bucket do Amazon S3 usando o Apache Flink 1.6.2, recomendamos o uso do Kinesis Data Firehose. Para obter mais informações sobre como usar o Kinesis Data Firehose com o Managed Service para Apache Flink, consulte Coletor do Kinesis Data Firehose.

O aplicativo está no status EM EXECUÇÃO, mas não está processando dados

Você pode verificar o status do seu aplicativo usando as ações `ListApplications` ou `DescribeApplication`. Se seu aplicativo inserir o status `RUNNING` mas não estiver gravando dados no coletor, você poderá solucionar o problema adicionando um fluxo de logs do Amazon CloudWatch ao seu aplicativo. Para obter mais informações, consulte Trabalhando com opções de CloudWatch registro de aplicativos. O fluxo de logs contém mensagens que podem ajudar a solucionar problemas do aplicativo.

Erro de instantâneo, atualização do aplicativo ou parada do aplicativo: InvalidApplicationConfigurationException

Um erro semelhante ao seguinte pode ocorrer durante uma operação de instantâneo ou durante uma operação que cria um instantâneo, como atualizar ou interromper um aplicativo:

```
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
```

Esse erro ocorre quando o aplicativo não consegue criar um snapshot.

Se você encontrar esse erro durante uma operação de instantâneo ou uma operação que cria um instantâneo, faça o seguinte:

- Desative os instantâneos do seu aplicativo. Você pode fazer isso no console do Managed Service for Apache Flink ou usando o parâmetro `SnapshotsEnabledUpdate` da ação UpdateApplication.
- Investigue por que os instantâneos não podem ser criados. Para obter mais informações, consulte O aplicativo está preso em um status transitório.
- Reative os instantâneos quando o aplicativo retornar a um estado íntegro.

java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

A localização do armazenamento confiável SSL foi atualizada em uma implantação anterior. Use o seguinte valor para o parâmetro `ssl.truststore.location`:

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

O aplicativo está sendo reiniciado

Se seu aplicativo não estiver íntegro, seu trabalho do Apache Flink falhará e será reiniciado continuamente. Esta seção descreve os sintomas e as etapas de solução de problemas dessa condição.

Sintomas

Essa condição pode ter os seguintes sintomas:

- A métrica `FullRestarts` não é zero. Essa métrica representa o número de vezes que o trabalho do aplicativo foi reiniciado desde que você iniciou o aplicativo.
- A métrica `Downtime` não é zero. Essa métrica representa o número de milissegundos em que o aplicativo está no status `FAILING` ou `RESTARTING`.
- O log do aplicativo contém alterações de status para `RESTARTING` ou `FAILED`. Você pode consultar o log do seu aplicativo para essas alterações de status usando a seguinte consulta do CloudWatch Logs Insights: Analise os erros: falhas relacionadas à tarefa do aplicativo.

Causas e soluções

As condições a seguir podem fazer com que seu aplicativo fique instável e seja reiniciado repetidamente:

- O operador está lançando uma exceção: se alguma exceção em um operador em seu aplicativo não for tratada, o aplicativo falhará (interpretando que a falha não pode ser tratada pelo operador). O aplicativo é reiniciado a partir do ponto de verificação mais recente para manter a semântica de processamento “exatamente uma vez”. Como resultado, o `Downtime` não é zero durante esses períodos de reinicialização. Para evitar que isso aconteça, recomendamos que você trate de quaisquer exceções que possam ser repetidas no código do aplicativo.

  Você pode investigar as causas dessa condição consultando os logs do aplicativo em busca de alterações no estado do aplicativo de `RUNNING` para `FAILED`. Para obter mais informações, consulte the section called “Analise os erros: falhas relacionadas à tarefa do aplicativo”.
- O Kinesis Data Streams não está devidamente provisionado: se uma fonte ou coletor do seu aplicativo for um fluxo de dados Kinesis, verifique se há erros `ReadProvisionedThroughputExceeded` ou `WriteProvisionedThroughputExceeded` nas métricas do fluxo.

  Se você encontrar esses erros, poderá aumentar a throughput disponível para o fluxo do Kinesis aumentando o número de fragmentos do fluxo. Para obter mais informações, consulte Como alterar o número de fragmentos abertos no Kinesis Data Streams?.
- Outras fontes ou coletores não estão adequadamente provisionados ou disponíveis: verifique se seu aplicativo está provisionando corretamente as fontes e coletores. Verifique se todas as fontes ou coletores usados no aplicativo (como outros serviços AWS ou fontes ou destinos externos) estão bem provisionados, não estão sofrendo limitação de leitura ou gravação no controle de utilização ou se estão periodicamente indisponíveis.

  Se você estiver enfrentando problemas relacionados à throughput com seus serviços dependentes, aumente os recursos disponíveis para esses serviços ou investigue a causa de quaisquer erros ou indisponibilidade.
- Os operadores não são provisionados adequadamente: se a workload nos threads de um dos operadores em seu aplicativo não for distribuída corretamente, o operador poderá ficar sobrecarregado e o aplicativo poderá falhar. Para obter informações sobre como ajustar o paralelismo do operador, consulte Gerencie o escalonamento do operador adequadamente.

- O aplicativo falha com DaemonException: esse erro aparece no log do aplicativo se você estiver usando uma versão do Apache Flink anterior à 1.11. Talvez seja necessário atualizar para uma versão posterior do Apache Flink para que uma versão KPL 0.14 ou posterior seja usada.
- O aplicativo falha com TimeoutException, FlinkException ou RemoteTransportException: esses erros podem aparecer no log do aplicativo se os gerenciadores de tarefas estiverem falhando. Se seu aplicativo estiver sobrecarregado, seus gerenciadores de tarefas podem sofrer pressão de recursos de CPU ou memória, fazendo com que falhem.

  Esses erros podem ter a seguinte aparência:

  - `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 essa condição, verifique o seguinte:

  - Verifique suas métricas do CloudWatch em busca de picos incomuns no uso da CPU ou da memória.
  - Verifique se há problemas de throughput em seu aplicativo. Para obter mais informações, consulte Solução de problemas de desempenho.
  - Examine o log do aplicativo em busca de exceções não tratadas que o código do aplicativo está gerando.
- O aplicativo falha com o erro JaxBannotationModule Not Found: esse erro ocorre se seu aplicativo usa o Apache Beam, mas não tem as dependências ou versões de dependência corretas. Os aplicativos do Managed Service for Apache Flink que usam o Apache Beam devem usar as seguintes versões de dependências:

```
<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>
```

Se você não fornecer a versão correta do `jackson-module-jaxb-annotations` como uma dependência explícita, seu aplicativo a carrega das dependências do ambiente e, como as versões não coincidem, o aplicativo falha no runtime.

Para obter informações sobre como usar o Apache Beam com o Managed Service para Apache Flink, consulte. Usar CloudFormation for Managed Service for Apache Flink

- O aplicativo falha com java.io.IOException: número insuficiente de buffers de rede

  Isso acontece quando um aplicativo não tem memória suficiente alocada para buffers de rede. Os buffers de rede facilitam a comunicação entre as subtarefas. Eles são usados para armazenar logs antes da transmissão por uma rede e para armazenar dados recebidos antes de dissecá-los em logs e entregá-los a subtarefas. O número de buffers de rede necessários é escalado diretamente com o paralelismo e a complexidade do seu gráfico de trabalho. Há várias abordagens para mitigar esse problema:

  - Você pode configurar um `parallelismPerKpu` menor para que haja mais memória alocada por subtarefa e buffers de rede. Observe que a redução de `parallelismPerKpu` aumentará a KPU e, portanto, o custo. Para evitar isso, você pode manter a mesma quantidade de KPU diminuindo o paralelismo pelo mesmo fator.
  - Você pode simplificar seu gráfico de tarefas reduzindo o número de operadores ou encadeando-os para que sejam necessários menos buffers.
  - Caso contrário, você pode acessar https://aws.amazon.com/premiumsupport/ para obter uma configuração personalizada do buffer de rede.

O throughput é muito lento

Se seu aplicativo não estiver processando os dados de transmissão recebidos com rapidez suficiente, ele terá um desempenho insatisfatório e ficará instável. Esta seção descreve os sintomas e as etapas de solução de problemas dessa condição.

Sintomas

Essa condição pode ter os seguintes sintomas:

- Se a fonte de dados do seu aplicativo for um fluxo do Kinesis, a métrica `millisbehindLatest` do fluxo aumentará continuamente.

- Se a fonte de dados do seu aplicativo for um cluster do Amazon MSK, as métricas de atraso do consumidor do cluster aumentam continuamente. Para obter mais informações, consulte Monitoramento de atraso do consumidor no Guia do desenvolvedor do Amazon MSK.
- Se a fonte de dados do seu aplicativo for um serviço ou fonte diferente, verifique todas as métricas de atraso do consumidor ou dados disponíveis.

Causas e soluções

Pode haver muitas causas para o throughput baixo do aplicativo. Se seu aplicativo não estiver acompanhando as entradas, verifique o seguinte:

- Se o atraso no throughput estiver aumentando e depois diminuindo, verifique se o aplicativo está sendo reiniciado. Seu aplicativo interromperá o processamento da entrada enquanto for reiniciado, causando um aumento no atraso. Para obter informações sobre falhas do aplicativo, consulte O aplicativo está sendo reiniciado.
- Se o atraso no throughput for consistente, verifique se seu aplicativo está otimizado para desempenho. Para obter informações sobre como otimizar o desempenho do seu aplicativo, consulte Solução de problemas de desempenho.
- Se o atraso no throughput não aumentar repentinamente, mas estiver aumentando continuamente, e seu aplicativo estiver otimizado para desempenho, você deverá aumentar os recursos do aplicativo. Para obter informações sobre o aumento dos recursos do aplicativo, consulte Escalabilidade.
- Se seu aplicativo lê de um cluster do Kafka em uma região diferente e `FlinkKafkaConsumer` ou `KafkaSource` está quase inativo (`idleTimeMsPerSecond` alto ou `CPUUtilization` baixo) apesar do alto atraso do consumidor, você pode aumentar o valor para `receive.buffer.byte`, como 2097152. Para obter mais informações, consulte a seção Ambiente de alta latência em Configurações personalizadas do MSK.

Para obter as etapas de solução de problemas relacionados ao throughput baixo ou ao aumento do atraso do consumidor na origem do aplicativo, consulte Solução de problemas de desempenho.

Crescimento ilimitado do estado

Se seu aplicativo não estiver descartando adequadamente as informações de estado desatualizadas, elas se acumularão continuamente e causarão problemas de desempenho ou estabilidade do aplicativo. Esta seção descreve os sintomas e as etapas de solução de problemas dessa condição.

Sintomas

Essa condição pode ter os seguintes sintomas:

- A `lastCheckpointDuration` métrica está aumentando ou aumentando gradualmente.
- A `lastCheckpointSize` métrica está aumentando ou aumentando gradualmente.

Causas e soluções

As condições a seguir podem fazer com que seu aplicativo acumule dados de estado:

- Seu aplicativo está retendo dados de estado por mais tempo do que o necessário.
- Seu aplicativo usa consultas de janela com uma duração muito longa.
- Você não definiu o TTL para os dados do seu estado. Para obter mais informações, consulte State Time-To-Live (TTL) na documentação do Apache Flink.
- Você está executando um aplicativo que depende da versão 2.25.0 ou mais recente do Apache Beam. Você pode optar por não participar da nova versão da transformação de leitura estendendo seu BeamApplicationProperties com os principais experimentos e valores `use_deprecated_read`. Para obter mais informações, consulte a documentação do Apache Sqoop.

Às vezes, os aplicativos enfrentam um crescimento cada vez maior do tamanho do estado, o que não é sustentável a longo prazo (afinal, um aplicativo Flink é executado indefinidamente). Às vezes, isso pode ser atribuído a aplicativos que armazenam dados no estado e não envelhecem adequadamente as informações antigas. Mas às vezes há expectativas irracionais sobre o que o Flink pode oferecer. Os aplicativos podem usar agregações em grandes janelas de tempo que abrangem dias ou até semanas. A menos que sejam usadas AggregateFunctions, que permitem agregações incrementais, o Flink precisa manter os eventos de toda a janela no estado.

Além disso, ao usar funções de processo para implementar operadores personalizados, o aplicativo precisa remover dados de um estado que não é mais necessário para a lógica de negócios. Nesse caso, o tempo de vida útil do estado pode ser usado para envelhecer automaticamente os dados com base no tempo de processamento. Managed Service for Apache Flink usa pontos de verificação incrementais e, portanto, o ttl de estado é baseado na compactação do RocksDB. Você só pode observar uma redução real no tamanho do estado (indicada pelo tamanho do ponto de verificação) após a ocorrência de uma operação de compactação. Em particular, para tamanhos de pontos de verificação abaixo de 200 MB, é improvável que você observe qualquer redução no tamanho dos

pontos de verificação como resultado da expiração do estado. No entanto, os pontos de salvamento são baseados em uma cópia limpa do estado que não contém dados antigos, portanto, você pode acionar um instantâneo no Managed Service for Apache Flink para forçar a remoção do estado desatualizado.

Para fins de depuração, pode fazer sentido desativar os pontos de verificação incrementais para verificar mais rapidamente se o tamanho do ponto de verificação realmente diminui ou se estabiliza (e evitar o efeito da compactação no RockSBS). No entanto, isso requer um tíquete para a equipe de serviço.

Operadores de E/S

É melhor evitar dependências de sistemas externos no caminho dos dados. Geralmente, é muito mais eficiente manter um conjunto de dados de referência em estado em vez de consultar um sistema externo para enriquecer eventos individuais. No entanto, às vezes há dependências que não podem ser facilmente transferidas para o estado, por exemplo, se você quiser enriquecer eventos com um modelo de machine learning hospedado no Amazon Sagemaker.

Os operadores que estão interagindo com sistemas externos pela rede podem se tornar um gargalo e causar contrapressão. É altamente recomendável usar o AsyncIO para implementar a funcionalidade, reduzir o tempo de espera de chamadas individuais e evitar que todo o aplicativo fique lento.

Além disso, para aplicativos com operadores vinculados a E/S, também pode fazer sentido aumentar a configuração de ParallelismPerKPU do aplicativo Managed Service for Apache Flink. Essa configuração descreve o número de subtarefas em paralelo que um aplicativo pode executar por unidade de processamento do Kinesis (KPU). Ao aumentar o valor do padrão de 1 para, digamos, 4, o aplicativo aproveita os mesmos recursos (e tem o mesmo custo), mas pode escalar até 4 vezes o paralelismo. Isso funciona bem para aplicativos vinculados a E/S, mas causa sobrecarga adicional para aplicativos que não estão vinculados a E/S.

Controle de utilização de fonte ou upstream de um fluxo de dados do Kinesis

Sintoma: o aplicativo detecta `LimitExceededExceptions` a partir da fonte upstream do fluxo de dados do Kinesis.

Causa potencial: a configuração padrão do conector Kinesis da biblioteca Apache Flink é definida para leitura da fonte do fluxo de dados do Kinesis com uma configuração padrão muito agressiva

para o número máximo de registros buscados por `GetRecords`chamada. O Apache Flink é configurado por padrão para buscar 10.000 registros por chamada GetRecords (essa chamada é feita por padrão a cada 200 ms), embora o limite por fragmento seja de apenas 1.000 registros.

Esse comportamento padrão pode levar à limitação ao tentar consumir do fluxo de dados do Kinesis, o que afetará o desempenho e a estabilidade dos aplicativos.

Isso pode ser confirmado verificando a métrica do `ReadProvisionedThroughputExceeded` CloudWatch e vendo períodos prolongados ou sustentados em que essa métrica é maior que zero.

O cliente também poderá ver isso nos logs do CloudWatch de seu aplicativo Managed Service for Apache Flink ao ver erros `LimitExceededException` contínuos.

Resolução: o cliente pode fazer uma das duas coisas para resolver esse cenário:

- Diminuir o limite padrão para o número de registros buscados por chamada de GetRecords
- O cliente pode habilitar leituras adaptáveis em seu aplicativo Managed Service for Apache Flink. Para obter mais informações sobre o recurso Adaptive Reads, consulte SHARD_USE_ADAPTIVE_READS

Pontos de verificação

Os pontos de verificação são o mecanismo do Flink para garantir que o estado de um aplicativo seja tolerante a falhas. O mecanismo permite que o Flink recupere o estado dos operadores se a tarefa falhar e fornece ao aplicativo a mesma semântica da execução sem falhas. Com o Managed Service for Apache Flink, o estado de um aplicativo é armazenado no RocksDB, um armazenamento de chave/valor incorporado que mantém seu estado de funcionamento no disco. Quando um ponto de verificação é usado, o estado também é carregado no Amazon S3. Portanto, mesmo que o disco seja perdido, o ponto de verificação pode ser usado para restaurar o estado do aplicativo.

Para obter mais informações, consulte Como o snapshot de estado funciona?.

Estágios do ponto de verificação

Para uma subtarefa de operador de ponto de verificação no Flink, há 5 estágios principais:

- Aguardando [Atraso inicial] — O Flink usa barreiras de ponto de verificação que são inseridas no fluxo, então o tempo neste estágio é o tempo em que o operador espera que a barreira do ponto de verificação chegue até ela.

- Alinhamento [Duração do alinhamento] — Nesse estágio, a subtarefa atingiu uma barreira, mas está aguardando barreiras de outros fluxos de entrada.
- Ponto de verificação de sincronização [Duração da sincronização] — Esse estágio é quando a subtarefa realmente captura o estado do operador e bloqueia todas as outras atividades na subtarefa.
- Ponto de verificação de assincronia [Duração da assincronia] — A maior parte desse estágio é a subtarefa de fazer o upload do estado para o Amazon S3. Durante esse estágio, a subtarefa não está mais bloqueada e pode processar registros.
- Confirmação — Geralmente é uma etapa curta e é simplesmente a subtarefa de enviar uma confirmação ao JobManager e também executar qualquer mensagem de confirmação (por exemplo, com coletores Kafka).

Cada um desses estágios (exceto Confirmação) é mapeado para uma métrica de duração para pontos de verificação que está disponível no Flink WebUI, o que pode ajudar a isolar a causa do longo ponto de verificação.

Para ver uma definição exata de cada uma das métricas disponíveis nos pontos de verificação, acesse a guia Histórico.

Investigar

Ao investigar a duração longa do ponto de verificação, a coisa mais importante a determinar é o gargalo do ponto de verificação, ou seja, qual operador e subtarefa estão demorando mais até o ponto de verificação e qual estágio dessa subtarefa está levando um período de tempo longo. Isso pode ser determinado usando o Flink WebUI na tarefa de ponto de verificação das tarefas. A interface web do Flink fornece dados e informações que ajudam a investigar problemas de ponto de verificação. Para obter uma análise completa, consulte Monitoramento de pontos de verificação.

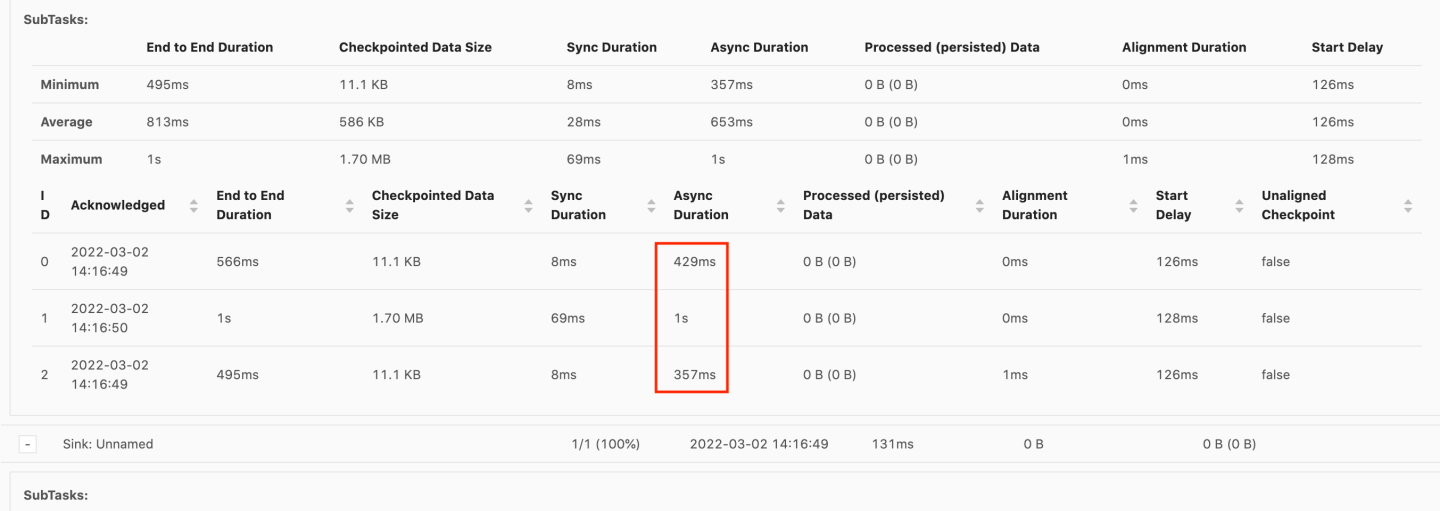
A primeira coisa a observar é a duração de ponta a ponta de cada operador no gráfico de tarefas para determinar qual operador está demorando até o ponto de verificação e merece uma investigação mais aprofundada. De acordo com a documentação do Flink, a definição da duração é:

A duração do timestamp do acionador até a confirmação mais recente (ou n/a, se nenhuma confirmação foi recebida ainda). Essa duração de ponta a ponta para um ponto de verificação completo é determinada pela última subtarefa que reconhece o ponto de verificação. Esse tempo geralmente é maior do que as subtarefas individuais necessárias para realmente verificar o estado.

As outras durações do ponto de verificação também fornecem informações mais detalhadas sobre onde o tempo está sendo gasto.

Se a Duração da sincronização for alta, isso indica que algo está acontecendo durante o instantâneo. Durante esse estágio, `snapshotState()` é chamado para classes que implementam a interface SnapshotState; isso pode ser um código de usuário, portanto, os despejos de thread podem ser úteis para investigar isso.

Uma Duração da assincronia longa sugere que muito tempo está sendo gasto no upload do estado para o Amazon S3. Isso pode ocorrer se o estado for grande ou se houver muitos arquivos de estado sendo carregados. Se esse for o caso, vale a pena investigar como o estado está sendo usado pelo aplicativo e garantir que as estruturas de dados nativas do Flink estejam sendo usadas sempre que possível (Uso do estado com chave). O Managed Service for Apache Flink configura o Flink de forma a minimizar o número de chamadas do Amazon S3 para garantir que não demore muito. A seguir há um exemplo das estatísticas de ponto de verificação de um operador. Isso mostra que a Duração da assincronia é relativamente longa em comparação com as estatísticas anteriores do ponto de verificação do operador.



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

- Sink: Unnamed 1/1 (100%) 2022-03-02 14:16:49 131ms 0 B 0 B (0 B)

SubTasks:

O Atraso inicial alto mostra que a maior parte do tempo está sendo gasta esperando que a barreira do ponto de verificação chegue ao operador. Isso indica que o aplicativo está demorando um pouco para processar os registros, o que significa que a barreira está fluindo lentamente pelo gráfico de tarefas. Geralmente, esse é o caso se a tarefa estiver sob contrapressão ou se um ou mais operadores estiverem constantemente ocupados. Veja a seguir um exemplo de um gráfico de tarefas em que o segundo operador KeyedProcess está ocupado.

Você pode investigar o que está demorando tanto usando os despejos de thread Flink Flame Graphs ou TaskManager thread dumps. Uma vez identificado o gargalo, ele pode ser investigado mais detalhadamente usando o Flame graphs ou despejos de thread.

Despejos de thread

Os despejos de thread são outra ferramenta de depuração que está em um nível um pouco mais baixo do que o Flame graphs. Um despejo de thread gera o estado de execução de todos os threads em um determinado momento. O Flink usa um despejo de thread JVM, que é um estado de execução de todos os threads no processo do Flink. O estado de um thread é apresentado por um rastreamento de pilha do thread, bem como por algumas informações adicionais. Na verdade, os Flame graphs são criados usando vários rastreamentos de pilha obtidos em rápida sucessão. O gráfico é uma visualização feita a partir desses traços que facilita a identificação dos caminhos de código comuns.

```
"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(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
    ...
```

Acima está um trecho de um despejo de thread retirado da interface do usuário do Flink para um único thread. A primeira linha contém algumas informações gerais sobre esse thread, incluindo:

- O nome do thread KeyedProcess (1/3) #0
- Prioridade do thread prio=5
- Uma ID de thread exclusiva Id=1423
- Estado do thread EXECUTÁVEL

O nome de um thread geralmente fornece informações sobre o propósito geral do thread. Os threads do operador podem ser identificados pelo nome, pois os threads do operador têm o mesmo nome do operador, bem como uma indicação da subtarefa à qual estão relacionados, por exemplo, o thread keyedProcess (1/3)#0 é do operador keyedProcess e da 1ª (de 3) subtarefa.

Os threads podem estar em um dos seguintes estados:

- NOVO — O thread foi criado, mas ainda não foi processado
- EXECUTÁVEL — O thread é executado na CPU
- BLOQUEADO — O thread está esperando que outro thread libere seu bloqueio
- AGUARDANDO — O thread está aguardando usando um método `wait()`, `join()` ou `park()`
- TEMPORIZADO_AGUARDANDO — O thread está aguardando usando um método sleep, wait, join ou park, mas com um tempo de espera máximo.

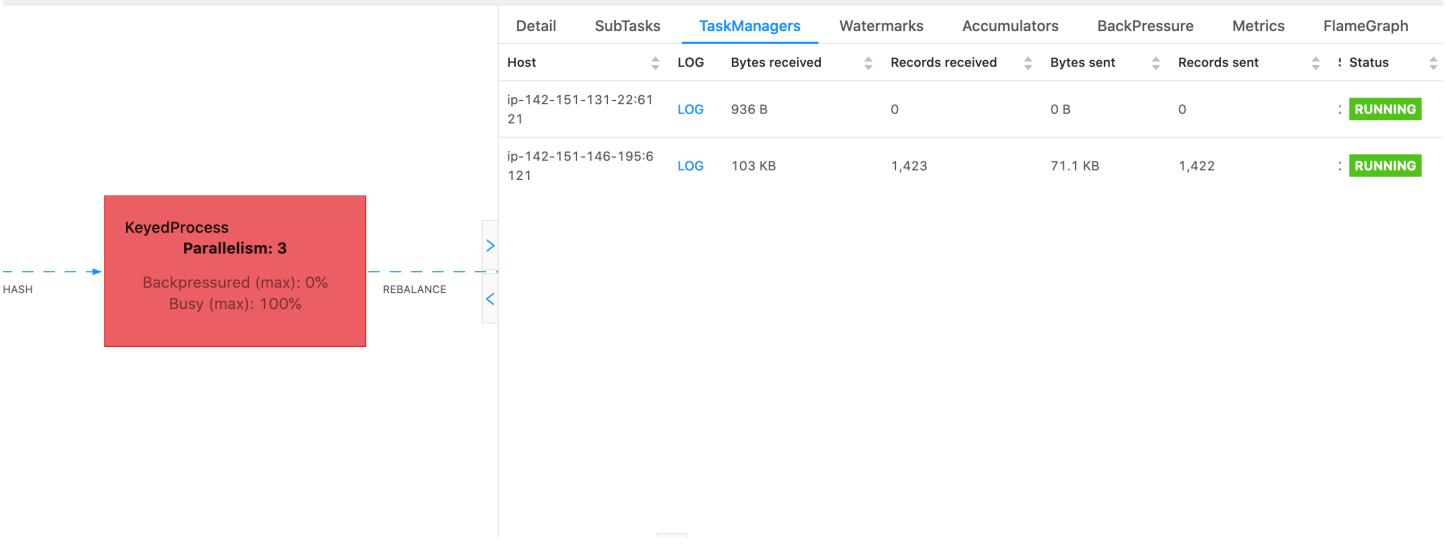
> **Note**
>
> No Flink 1.13, a profundidade máxima de um único rastreamento de pilha no despejo de thread é limitada a 8.

> **Note**
>
> Os despejos de thread devem ser o último recurso para depurar problemas de desempenho em um aplicativo Flink, pois podem ser difíceis de ler e exigem que várias amostras sejam coletadas e analisadas manualmente. Se possível, é preferível usar Flame graphs.

Despejos de thread no Flink

No Flink, um despejo de thread pode ser feito escolhendo a opção Gerenciadores de tarefas na barra de navegação esquerda da interface do usuário do Flink, selecionando um gerenciador de tarefas específico e navegando até a guia Despejo de thread. O despejo de thread pode ser baixado, copiado para seu editor de texto (ou analisador de thread dump) favorito ou analisado diretamente na visualização de texto na interface do usuário Web do Flink (no entanto, essa última opção pode ser um pouco complicada.

Para determinar qual Gerenciador de tarefas usar, um despejo de thread da guia Gerenciadores de tarefas pode ser usado quando um determinado operador é escolhido. Isso mostra que o operador está sendo executado em diferentes subtarefas de um operador e pode ser executado em diferentes gerenciadores de tarefas.



O dump será composto por vários rastreamentos de pilha. No entanto, ao investigar o dump, os relacionados a um operador são os mais importantes. Eles podem ser facilmente encontrados, pois os threads do operador têm o mesmo nome do operador, bem como uma indicação da subtarefa à qual estão relacionados. Por exemplo, o rastreamento de pilha a seguir é do operador KeyedProcess e é a primeira subtarefa.

```
"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
    ...
```

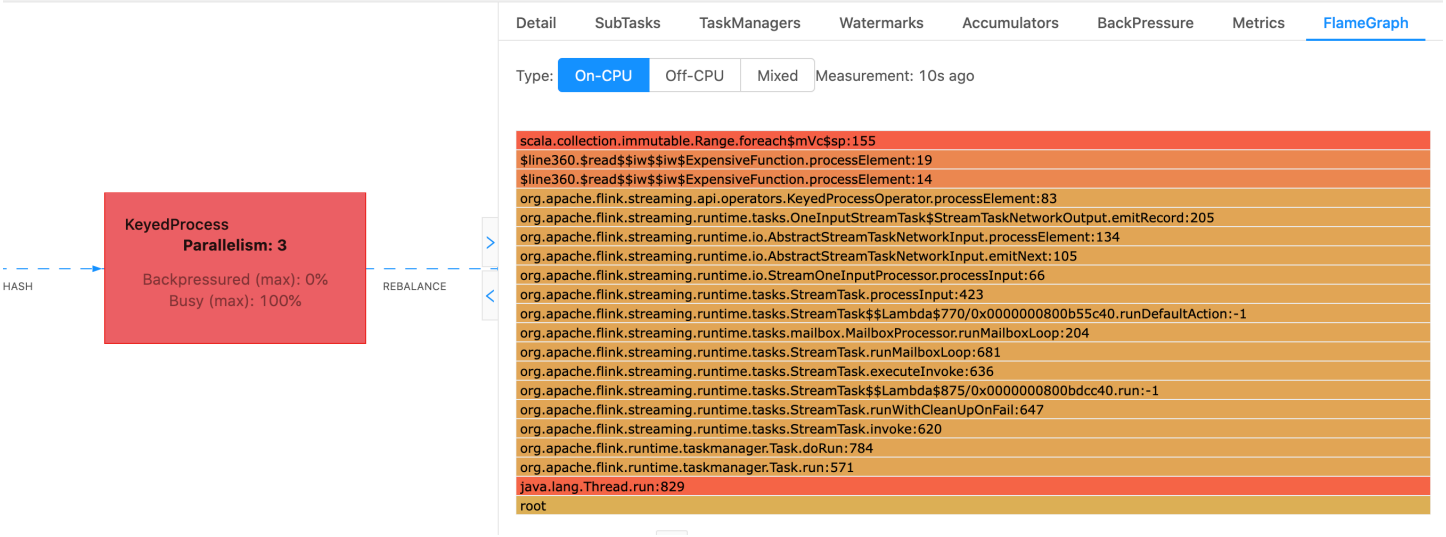
Isso pode se tornar confuso se houver vários operadores com o mesmo nome, mas podemos nomear operadores para contornar isso. Por exemplo:

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

Flame graphs

Os Flame graphs são uma ferramenta de depuração útil que visualiza os rastreamentos da pilha do código de destino, o que permite identificar os caminhos de código mais frequentes. Eles são criados por meio de rastreamentos de pilha de amostras várias vezes. O eixo x de um Flame graph mostra os diferentes perfis da pilha, enquanto o eixo y mostra a profundidade da pilha e chama o rastreamento da pilha. Um único retângulo em um Flame graph representa uma estrutura de pilha, e a largura de uma chama mostra com que frequência ela aparece nas pilhas. Para obter mais detalhes sobre os Flame graphs e como usá-los, consulte Flame graphs.

No Flink, o Flame graph de um operador pode ser acessado por meio da interface do usuário Web selecionando um operador e, em seguida, escolhendo a guia Flame graph. Depois que amostras suficientes forem coletadas, o Flame graph será exibido. A seguir está o Flame graph para a função de processo que estava demorando muito para ser verificada.



Este é um Flame graph muito simples e mostra que todo o tempo da CPU está sendo gasto em uma visualização foreach no `processElement` do operador ExpensiveFunction. Você também obtém o número da linha para ajudar a determinar onde a execução do código está ocorrendo.

O ponto de verificação está atingindo o tempo limite

Se seu aplicativo não for otimizado ou provisionado adequadamente, os pontos de verificação podem falhar. Esta seção descreve os sintomas e as etapas de solução de problemas dessa condição.

Sintomas

Se os pontos de verificação falharem em seu aplicativo, o `numberOfFailedCheckpoints` será maior que zero.

Os pontos de verificação podem falhar devido a falhas diretas, como erros do aplicativo, ou devido a falhas transitórias, como a falta de recursos do aplicativo. Verifique os logs e as métricas do seu aplicativo para ver os seguintes sintomas:

- Erros no seu código.
- Erros ao acessar os serviços dependentes do seu aplicativo.
- Erros ao serializar dados. Se o serializador padrão não conseguir serializar os dados do aplicativo, o aplicativo falhará. Para obter informações sobre como usar um serializador personalizado em seu aplicativo, consulte Serializadores personalizados na Documentação do Apache Flink.
- Erros de falta de memória.

- Picos ou aumentos constantes nas seguintes métricas:
  - `heapMemoryUtilization`
  - `oldGenerationGCTime`
  - `oldGenerationGCCount`
  - `lastCheckpointSize`
  - `lastCheckpointDuration`

Para obter mais informações sobre o monitoramento de pontos de verificação, consulte Monitoramento de pontos de verificação na Documentação do Apache Flink.

Causas e soluções

As mensagens de erro de log do aplicativo mostram a causa das falhas diretas. Falhas transitórias podem ter as seguintes causas:

- Seu aplicativo tem provisionamento de KPU insuficiente. Para obter informações sobre como aumentar o provisionamento de aplicativos, consulte Escalabilidade.
- O tamanho do estado do seu aplicativo é muito grande. Você pode monitorar o tamanho do estado do seu aplicativo usando a métrica `lastCheckpointSize`.
- Os dados de estado do seu aplicativo são distribuídos de forma desigual entre as chaves. Se seu aplicativo usa o operador `KeyBy`, verifique se os dados recebidos estão sendo divididos igualmente entre as chaves. Se a maioria dos dados estiver sendo atribuída a uma única chave, isso cria um gargalo que causa falhas.
- Seu aplicativo está enfrentando contrapressão na memória ou na coleta de resíduos. Monitore `heapMemoryUtilization`, `oldGenerationGCTime` e `oldGenerationGCCount` do seu aplicativo em busca de picos ou valores cada vez maiores.

Falha no ponto de verificação do aplicativo Apache Beam

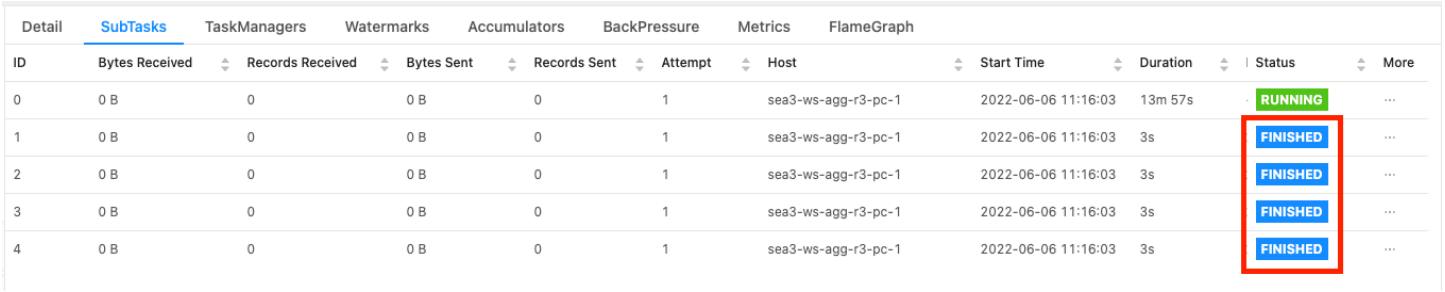
Se seu aplicativo Beam estiver configurado com shutdownSourcesAfterIdlems definido como 0ms, os pontos de verificação podem falhar ao serem acionados porque as tarefas estão no estado “FINALIZADO”. Esta seção descreve os sintomas e a resolução dessa condição.

Sintomas

Acesse os logs do CloudWatch do aplicativo Managed Service for Apache Flink e verifique se a seguinte mensagem de log foi registrada. A mensagem de log a seguir indica que o ponto de verificação falhou ao ser acionado, pois algumas tarefas foram concluídas.

```
                {
                "locationInformation":
"org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordina
                "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"
                }
```

Isso também pode ser encontrado no painel do Flink, onde algumas tarefas entraram no estado “CONCLUÍDO” e o ponto de verificação não é mais possível.



Causa

ShutdownSourcesAfterIdlems é uma variável de configuração do Beam que desliga fontes que ficaram inativas pelo tempo configurado de milissegundos. Depois que uma fonte é desligada, o ponto de verificação não é mais possível. Isso pode levar à falha do ponto de verificação.

Uma das causas para as tarefas entrarem no estado “FINALIZADO” é quando ShutdownSourcesAfterIdlems é definido como 0ms, o que significa que as tarefas que estão ociosas serão encerradas imediatamente.

Solução

Para evitar que as tarefas entrem imediatamente no estado “FINALIZADO”, defina ShutdownSourcesAfterIdlems como Long.MAX_VALUE. Isso pode ser feito de duas maneiras:

- Opção 1: Se a configuração do Beam estiver definida na página de configuração do aplicativo Managed Service for Apache Flink, você poderá adicionar um novo par de valores-chave para definir ShutdPwnSourcesAfterIdlems da seguinte forma:

- Opção 2: Se a configuração do Beam estiver definida no arquivo JAR, você poderá definir ShutdownSourcesAfterIdlems da seguinte forma:

```
                    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
```

Contrapressão

O Flink usa contrapressão para adaptar a velocidade de processamento de operadores individuais.

O operador pode ter dificuldade em continuar processando o volume de mensagens que recebe por vários motivos. A operação pode exigir mais recursos de CPU do que o operador tem disponíveis. O operador pode esperar que as operações de E/S sejam concluídas. Se um operador não consegue processar eventos com rapidez suficiente, isso gera contrapressão nos operadores a montante, alimentando o operador lento. Isso faz com que os operadores a montante diminuam a velocidade, o que pode propagar ainda mais a contrapressão para a fonte e fazer com que a fonte se adapte ao

throughput geral do aplicativo, diminuindo também a velocidade. Você pode encontrar uma descrição mais detalhada da contrapressão e de como ela funciona em Como o Apache Flink™ lida com a contrapressão.

Saber quais operadores em um aplicativo são lentos fornece informações cruciais para entender a causa raiz dos problemas de desempenho no aplicativo. As informações de contrapressão são expostas por meio do painel do Flink. Para identificar o operador lento, procure o operador com um valor alto de contrapressão que esteja mais próximo de um coletor (operador B no exemplo a seguir). O operador que causa a lentidão é então um dos operadores a jusante (operador C no exemplo). O B pode processar eventos mais rapidamente, mas sofre contrapressão, pois não pode encaminhar a saída para o verdadeiro operador lento C.

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D
 (backpressured 0%)
```

Depois de identificar o operador lento, tente entender por que ele é lento. Pode haver uma infinidade de motivos e, às vezes, não é óbvio o que está errado e pode exigir dias de depuração e criação de perfil para ser resolvido. A seguir estão alguns motivos óbvios e mais comuns, alguns dos quais são explicados mais detalhadamente abaixo:

- O operador está fazendo E/S lentas, por exemplo, chamadas de rede (considere usar o AsyncIO em vez disso).
- Há uma distorção nos dados, e um operador está recebendo mais eventos do que outros (verifique observando o número de mensagens de entrada/saída de subtarefas individuais (ou seja, instâncias do mesmo operador) no painel do Flink.
- É uma operação que consome muitos recursos (se não houver distorção de dados, considere aumentar a escala horizontalmente para o trabalho vinculado à CPU/memória ou aumentar `ParallelismPerKPU` para o trabalho vinculado à E/S)
- Registro de logs extensivo no operador (reduza o registro de logs ao mínimo para o aplicativo de produção ou considere enviar a saída de depuração para um fluxo de dados).

Testando o throughput com o coletor de descarte

O Coletor de descarte simplesmente ignora todos os eventos que recebe enquanto ainda executa o aplicativo (um aplicativo sem nenhum coletor falha na execução). Isso é muito útil para testes de throughput, criação de perfis e para verificar se o aplicativo está escalando adequadamente. Também é uma verificação de integridade muito pragmática para verificar se os coletores estão

causando contrapressão ou o aplicativo (mas verificar apenas as métricas de contrapressão geralmente é mais fácil e direto).

Ao substituir todos os coletores de um aplicativo por um coletor de descarte e criar uma fonte simulada que gera dados que se assemelham aos dados de produção, você pode medir o throughput máximo do aplicativo para uma determinada configuração de paralelismo. Em seguida, você também pode aumentar o paralelismo para verificar se o aplicativo está escalando adequadamente e não tem um gargalo que só surge com um throughput mais alto (por exemplo, devido à distorção de dados).

Distorção de dados

Um aplicativo Flink é executado em um cluster de forma distribuída. Para aumentar a escala horizontalmente para vários nós, o Flink usa o conceito de fluxos com chave, o que significa essencialmente que os eventos de um fluxo são particionados de acordo com uma chave específica, por exemplo, ID do cliente, e o Flink pode então processar partições diferentes em nós diferentes. Muitos dos operadores do Flink então são avaliados com base nessas partições, por exemplo, janelas com chave, funções de processo e E/S assíncrona.

A escolha de uma chave de partição geralmente depende da lógica de negócios. Ao mesmo tempo, muitas das melhores práticas para, por exemplo, DynamoDB e Spark, se aplicam igualmente ao Flink, incluindo:

- garantir uma alta cardinalidade das chaves de partição
- evitar distorções no volume de eventos entre as partições

Você pode identificar distorções nas partições comparando os registros recebidos/enviados de subtarefas (ou seja, instâncias do mesmo operador) no painel do Flink. Além disso, o monitoramento do Managed Service for Apache Flink também pode ser configurado para expor métricas para `numRecordsIn/Out` e `numRecordsInPerSecond/OutPerSecond` em um nível de subtarefa.

Distorção de estado

Para operadores com estado, ou seja, operadores que mantêm o estado de sua lógica de negócios, como janelas, a distorção de dados sempre leva à distorção de estado. Algumas subtarefas recebem mais eventos do que outras devido à distorção nos dados e, portanto, também persistem mais dados no estado. No entanto, mesmo para um aplicativo que tenha partições balanceadas uniformemente, pode haver uma distorção na quantidade de dados persistentes no estado. Por exemplo, para

janelas de sessão, alguns usuários e sessões, respectivamente, podem ser muito mais longos do que outros. Se as sessões mais longas fizerem parte da mesma partição, isso pode levar a um desequilíbrio do tamanho do estado mantido por diferentes subtarefas do mesmo operador.

A distorção de estado não apenas aumenta mais recursos de memória e disco exigidos por subtarefas individuais, mas também pode diminuir o desempenho geral do aplicativo. Quando um aplicativo está passando por um ponto de verificação ou ponto de salvamento, o estado do operador persiste no Amazon S3, para proteger o estado contra falhas no nó ou no cluster. Durante esse processo (especialmente com exatamente uma semântica ativada por padrão no Managed Service for Apache Flink), o processamento é interrompido de uma perspectiva externa até que o ponto de verificação/ponto de salvamento seja concluído. Se houver distorção de dados, o tempo para concluir a operação pode ser limitado por uma única subtarefa que tenha acumulado uma quantidade particularmente alta de estado. Em casos extremos, a obtenção de pontos de verificação/pontos de salvamento pode falhar porque uma única subtarefa não consegue persistir no estado.

Assim como a distorção de dados, a distorção de estado pode reduzir substancialmente a velocidade de um aplicativo.

Para identificar a distorção de estado, você pode aproveitar o painel do Flink. Encontre um ponto de verificação ou ponto de salvamento recente e compare a quantidade de dados que foram armazenados para subtarefas individuais nos detalhes.

Integração com recursos em diferentes regiões

Você pode habilitar usando `StreamingFileSink` para gravar em um bucket do Amazon S3 em uma região diferente do seu aplicativo Managed Service for Apache Flink por meio de uma configuração necessária para replicação entre regiões na configuração do Flink. Para fazer isso, registre um ticket de suporte no AWS SupportCentro.

Histórico de documentos do Amazon Managed Service for Apache Flink

A tabela a seguir descreve as alterações importantes na documentação desde a última versão do Managed Service for Apache Flink.

- Versão da API: 23/05/2018
- Última atualização da documentação: 30 de agosto de 2023

Alteração	Descrição	Data
O Kinesis Data Analytics é agora conhecido como Managed Service for Apache Flink	Não há alterações nos endpoints de serviço, nas APIs, na interface de linha de comando, nas políticas de acesso do IAM, nas CloudWatch métricas ou nos painéis de AWS faturamento. Seus aplicativos existentes continuarão a funcionar como antes. Para obter mais informações, consulte O que é Managed Service for Apache Flink?	30 de agosto de 2023
Suporte ao Apache Flink versão 1.15.2	O Managed Service for Apache Flink agora é compatível com aplicativos que usam o Apache Flink versão 1.15.2. Crie aplicativos do Kinesis Data Analytics usando a API de tabela Apache Flink. Para ter mais informações, consulte Criar aplicativos.	22 de novembro de 2022

Alteração	Descrição	Data
Suporte ao Apache Flink versão 1.13.2	O Managed Service for Apache Flink agora é compatível com aplicativos que usam o Apache Flink versão 1.13.2. Crie aplicativos do Kinesis Data Analytics usando a API de tabela Apache Flink. Para ter mais informações, consulte Introdução: Flink 1.13.2.	13 de outubro de 2021
Suporte ao Python	O Managed Service for Apache Flink agora é compatível com aplicativos que usam Python com a API de tabela e SQL do Apache Flink. Para ter mais informações, consulte Usar o Python.	25 de março de 2021
Suporte ao Apache Flink versão 1.11.1	O Managed Service for Apache Flink agora é compatível com aplicativos que usam o Apache Flink versão 1.11.1. Crie aplicativos do Kinesis Data Analytics usando a API de tabela Apache Flink. Para ter mais informações, consulte Criar aplicativos.	19 de novembro de 2020

Alteração	Descrição	Data
Painel do Apache Flink	Use o painel do Apache Flink para monitorar a integridade e o desempenho do aplicativo. Para ter mais informações, consulte Painel do Apache Flink.	19 de novembro de 2020
Consumidor EFO	Crie aplicativos que usem um consumidor de Enhanced Fan-Out (EFO) para ler de um fluxo de dados Kinesis. Para ter mais informações, consulte Consumidor EFO.	6 de outubro de 2020
Apache Beam	Crie aplicativos que usam o Apache Beam para processar dados de transmissão. Para ter mais informações, consulte Usar CloudFormation for Managed Service for Apache Flink.	15 de setembro de 2020
Performance	Como solucionar problemas de desempenho do aplicativo e como criar um aplicativo com desempenho. Para ter mais informações, consulte Desempenho.	21 de julho de 2020

Alteração	Descrição	Data
Armazenamento de chaves personalizado	Como acessar um cluster do Amazon MSK que usa um armazenamento de chaves personalizado para criptografia em trânsito. Para ter mais informações, consulte Truststore personalizado.	10 de junho de 2020
CloudWatch Alarmes	Recomendações para criar CloudWatch alarmes com o Managed Service for Apache Flink. Para ter mais informações, consulte Alarmes.	5 de junho de 2020
Novas CloudWatch métricas	O Managed Service for Apache Flink agora emite 22 métricas para a Amazon Metrics. CloudWatch Para ter mais informações, consulte Métricas e dimensões no Managed Service for Apache Flink.	12 de maio de 2020
CloudWatch Métricas personalizadas	Defina métricas específicas do aplicativo e as emita para a Amazon Metrics. CloudWatch Para ter mais informações, consulte Métricas personalizadas.	12 de maio de 2020

Alteração	Descrição	Data
Exemplo: Leia a partir de um fluxo de dados do Kinesis em outra conta.	Saiba como acessar uma transmissão do Kinesis em uma conta AWS diferente em seu aplicativo Managed Service for Apache Flink. Para ter mais informações, consulte Entre contas.	30 de março de 2020
Suporte ao Apache Flink versão 1.8.2	O Managed Service for Apache Flink agora é compatível com aplicativos que usam o Apache Flink versão 1.8.2. Use o Streaming FileSink conector Flink para gravar a saída diretamente no S3. Para ter mais informaçõ es, consulte Criar aplicativos.	17 de dezembro de 2019
VPC do Managed Service for Apache Flink	Configure um aplicativo do Managed Service for Apache Flink para se conectar a uma nuvem privada virtual. Para ter mais informações, consulte Usar uma Amazon VPC.	25 de novembro de 2019
Melhores práticas do Managed Service for Apache Flink	Melhores práticas para criar e administrar o Managed Service for Apache Flink. Para ter mais informações, consulte Práticas recomendadas.	14 de outubro de 2019

Alteração	Descrição	Data
Analisar logs de aplicativo do Managed Service for Apache Flink	Use o CloudWatch Logs Insights para monitorar seu serviço gerenciado para o aplicativo Apache Flink. Para ter mais informações, consulte Analisando logs.	26 de junho de 2019
Propriedades de runtime do aplicativo do Managed Service for Apache Flink	Trabalhe com as proprieda des de runtime no Managed Service for Apache Flink. Para ter mais informações, consulte Propriedades de runtime.	24 de junho de 2019
Marcação de aplicativos do Managed Service for Apache Flink	Use a marcação de aplicativos para determinar os custos por aplicativo, controlar o acesso ou para finalidades definidas pelo usuário. Para ter mais informações, consulte Uso de tags.	8 de maio de 2019
Exemplos de aplicativos do Managed Service for Apache Flink	Exemplos de aplicativos do Managed Service for Apache Flink demonstrando operadore s de janela e gravando a saída em Logs. CloudWatch Para ter mais informações, consulte Exemplos.	1º de maio de 2019

Alteração	Descrição	Data
Registrando em log as chamadas de API para o Managed Service for Apache Flink com AWS CloudTrail	O Managed Service for Apache Flink é integrado ao AWS CloudTrail, serviço que fornece um registro das ações executadas por um usuário, função ou serviço AWS no Managed Service for Apache Flink. Para ter mais informaçõ es, consulte Usando o AWS CloudTrail.	22 de março de 2019
Criar um aplicativo (coletor do Kinesis Data Firehose)	Faça um exercício de criar um Managed Service for Apache Flink com um fluxo de dados do Amazon Kinesis como fonte e um fluxo do Amazon Kinesis Data Firehose como coletor. Para ter mais informações, consulte Coletor do Kinesis Data Firehose.	13 de dezembro de 2018
Versão pública	Este é o lançamento inicial do Guia do desenvolvedor do Managed Service for Apache Flink para aplicativos Java.	27 de novembro de 2018

Exemplo de código de API para o Managed Service for Apache Flink

Este tópico contém exemplos de blocos de solicitação para ações no Managed Service for Apache Flink.

Para usar JSON como entrada para uma ação com o AWS Command Line Interface (AWS CLI), salve a solicitação em um arquivo JSON. Em seguida, passe o nome do arquivo para a ação usando o parâmetro `--cli-input-json`.

O exemplo a seguir demonstra como usar um arquivo JSON com uma ação.

```
$ aws kinesisanalyticsv2 start-application --cli-input-json file://start.json
```

Para obter mais informações sobre como usar o JSON com AWS CLI, consulte Gerar parâmetros JSON do esqueleto da CLI e da entrada da CLI no Guia do usuário AWS Command Line Interface.

Tópicos

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- AddApplicationVpcConfiguration
- CreateApplication
- CreateApplicationSnapshot
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource
- DeleteApplicationSnapshot

- DeleteApplicationVpcConfiguration
- DescribeApplication
- DescribeApplicationSnapshot
- DiscoverInputSchema
- ListApplications
- ListApplicationSnapshots
- StartApplication
- StopApplication
- UpdateApplication

AddApplicationCloudWatchLoggingOption

O exemplo de código de solicitação a seguir para a ação AddApplicationCloudWatchLoggingOption adiciona uma opção de Amazon CloudWatch Logs a um aplicativo do Managed Service for 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

O exemplo de código de solicitação a seguir para a ação AddApplicationInput adiciona uma entrada de aplicativo ao aplicativo do Managed Service for 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

O exemplo de código de solicitação a seguir para a ação AddApplicationInputProcessingConfiguration adiciona uma configuração de processamento de entrada de aplicativo ao aplicativo do Managed Service for Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "InputId": "2.1",
   "InputProcessingConfiguration": {
```

```
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   }
}
```

AddApplicationOutput

O exemplo de código de solicitação a seguir para a ação AddApplicationOutput adiciona o fluxo de dados Kinesis como uma saída de aplicativo ao aplicativo do Managed Service for 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

O exemplo de código de solicitação a seguir para a ação AddApplicationReferenceDataSource adiciona uma fonte de dados de referência do aplicativo CVS ao aplicativo do Managed Service for 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

O exemplo de código de solicitação a seguir para a ação AddApplicationVpcConfiguration adiciona uma configuração de VPC a uma aplicativo existente:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

O exemplo de código de solicitação a seguir para a ação CreateApplication cria um aplicativo do 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

O exemplo de código de solicitação a seguir para a ação CreateApplicationSnapshot cria um snapshot do estado do aplicativo:

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

DeleteApplication

O exemplo de código de solicitação a seguir para a ação DeleteApplication exclui um aplicativo do Managed Service para Apache Flink:

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

O exemplo de código de solicitação a seguir para a ação DeleteApplicationCloudWatchLoggingOption exclui uma opção do Amazon CloudWatch Log do Managed Service para Apache Flink:

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

```
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

O exemplo de código de solicitação a seguir para a ação DeleteApplicationInputProcessingConfiguration exclui uma configuração de processamento de entrada de um aplicativo do Managed Service para Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

O exemplo de código de solicitação a seguir para a ação DeleteApplicationOutput exclui uma saída de aplicativo de um aplicativo do Managed Service para Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

O exemplo de código de solicitação a seguir para a ação DeleteApplicationReferenceDataSource exclui uma fonte de dados de referência de aplicativo de um aplicativo do Managed Service para Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

O exemplo de código de solicitação a seguir para a ação DeleteApplicationSnapshot exclui um snapshot do estado do aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

O exemplo de código de solicitação a seguir para a ação DeleteApplicationVpcConfiguration exclui uma configuração de VPC existente de um aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

DescribeApplication

O exemplo de código de solicitação a seguir para a ação DescribeApplication apresenta detalhes sobre um aplicativo do Managed Service para Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

O exemplo de código de solicitação a seguir para a ação DescribeApplicationSnapshot apresenta detalhes sobre um snapshot de estado de aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
```

```
}
```

DiscoverInputSchema

O exemplo de código de solicitação a seguir para a ação DiscoverInputSchema gera um esquema de uma fonte de transmissão:

```
{
   "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"
}
```

O exemplo de código de solicitação a seguir para a ação DiscoverInputSchema gera um esquema de uma fonte de referência:

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

O exemplo de código de solicitação a seguir para a ação ListApplications retorna uma lista de aplicativos do Managed Service for Apache Flink em sua conta:

```
{
   "ExclusiveStartApplicationName": "MyApplication",
   "Limit": 50
}
```

ListApplicationSnapshots

O exemplo de código de solicitação a seguir para a ação ListApplicationSnapshots retorna uma lista de snapshots do estado do aplicativo:

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

O exemplo de código de solicitação a seguir para a ação StartApplication inicia um aplicativo Managed Service for Apache Flink e carrega o estado do aplicativo a partir do snapshot mais recente (se houver):

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

StopApplication

O exemplo de código de solicitação a seguir para a ação API_StopApplication interrompe um aplicativo do Managed Service para Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

O exemplo de código de solicitação a seguir para a ação UpdateApplication atualiza um aplicativo Managed Service for Apache Flink para alterar a localização do código do aplicativo:

```
{"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
      "ApplicationCodeConfigurationUpdate": {
         "CodeContentTypeUpdate": "ZIPFILE",
         "CodeContentUpdate": {
            "S3ContentLocationUpdate": {
               "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
               "FileKeyUpdate": "my_new_code.zip",
               "ObjectVersionUpdate": "2"
         }
      }
   }
}
```

Referência de APIs do Managed Service for Apache Flink

Para obter informações sobre as APIs que o Managed Service for Apache Flink fornece, consulte Referência de APIs do Managed Service for Apache Flink.