aws

Guia do desenvolvedor do Managed Service for Apache Flink

Managed Service for Apache Flink

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

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 o Amazon Managed Service para 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.

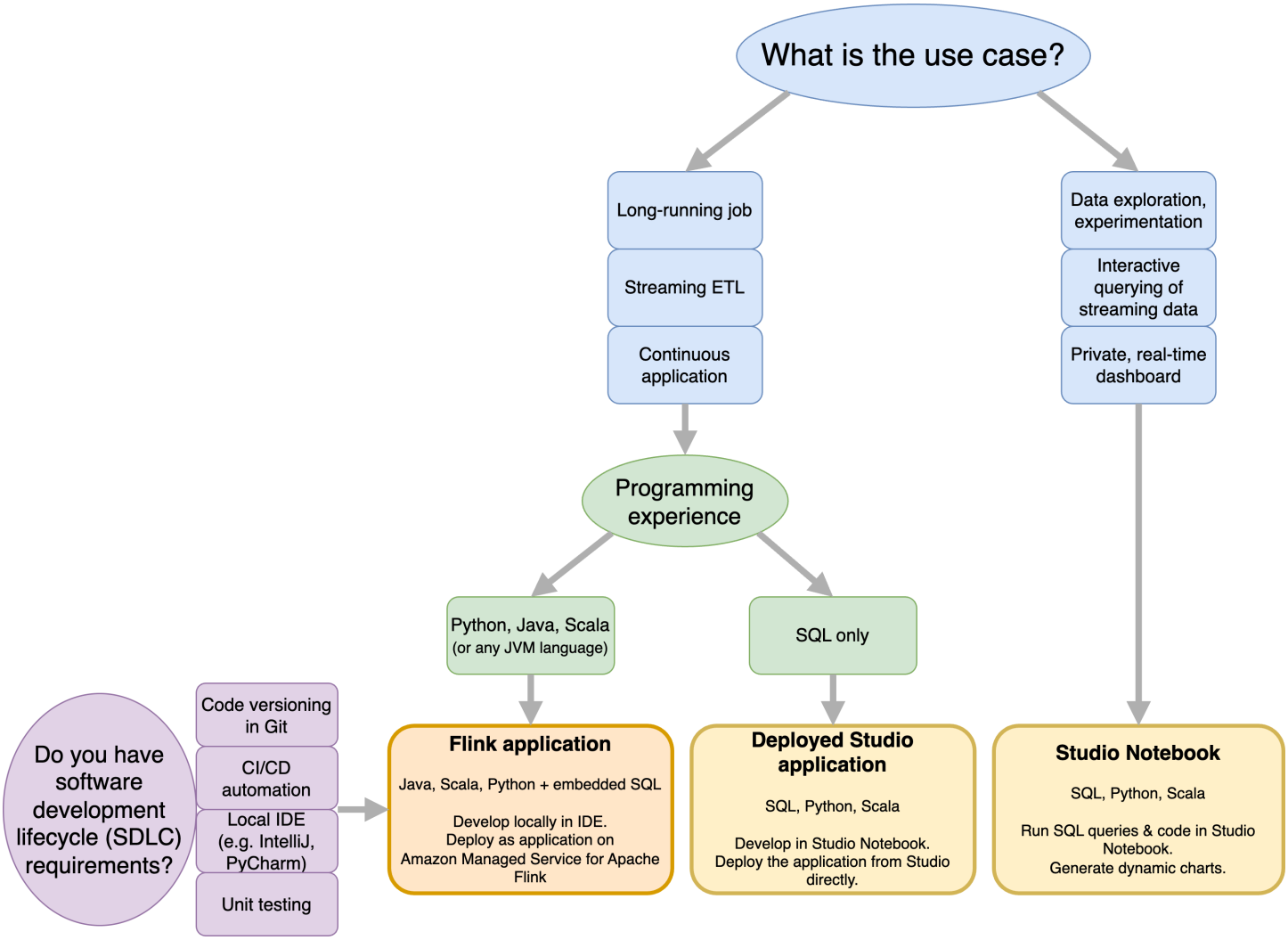
Você pode criar aplicativos com a linguagem de sua escolha no Managed Service for Apache Flink usando bibliotecas de código aberto baseadas no Apache Flink. O Apache Flink é uma estrutura popular e um mecanismo para o processamento de fluxos de dados.

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.

Decida entre usar o Managed Service para Apache Flink ou o Managed Service para 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 incorporadosSQL) usando um IDE de sua escolha e o Apache Flink Datastream ou Table. APIs 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 padrões, SQL Python e Scala.

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 aplicativos contínuos, considere usar o Managed Service for Apache Flink. Isso ocorre porque você pode criar seu aplicativo Flink usando o Flink APIs diretamente no aplicativo IDE de sua escolha. Desenvolver localmente com você 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 isso SQL podem considerar a implantação de um aplicativo de longa duração 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 SDLC ferramentas, como controle de versão de código no Git e automação de CI/CD, ou técnicas como testes unitários, recomendamos o Managed Service for Apache Flink usando o de sua escolha. IDE

Escolha qual Apache Flink usar no Managed Service APIs para Apache Flink

Você pode criar aplicativos usando Java, Python e Scala no Managed Service for Apache Flink usando o Apache Flink em um de sua escolha. APIs IDE Você pode encontrar orientações sobre como criar aplicativos usando o Flink Datastream and Table API na documentação. Você pode selecionar o idioma no qual você cria seu aplicativo Flink e o 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.

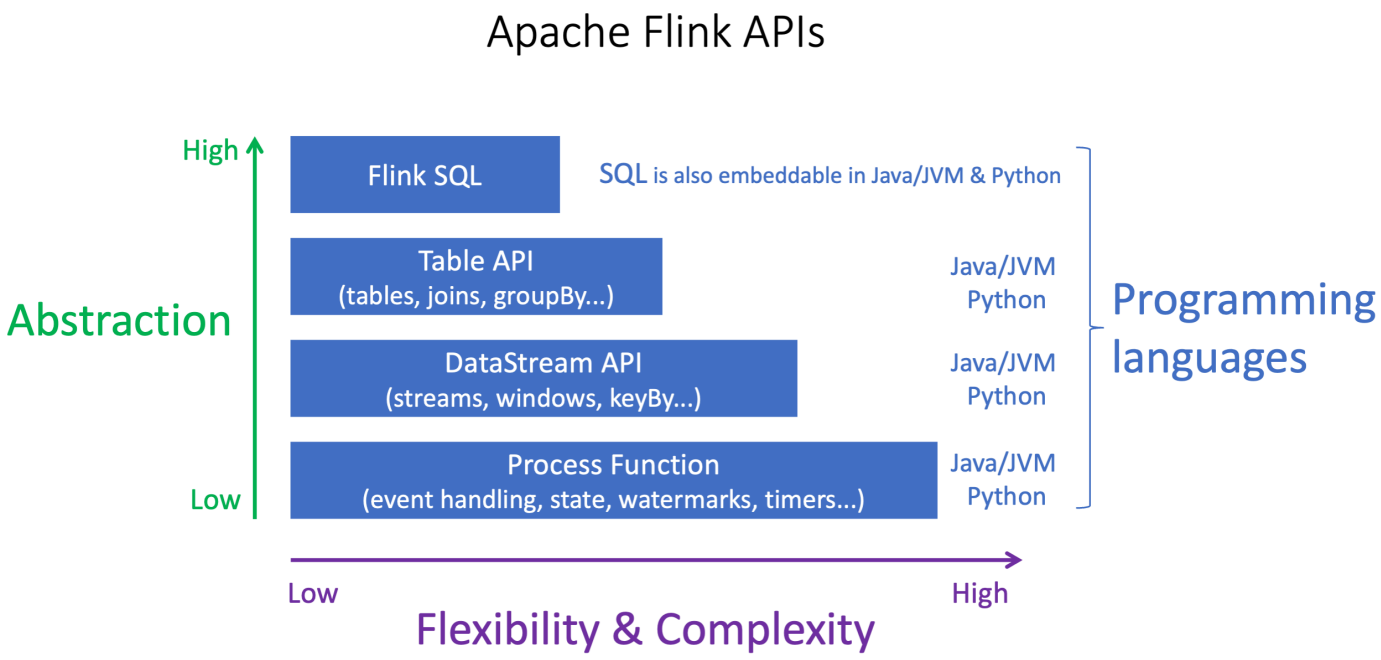
Escolha um Flink API

O Apache Flink APIs tem 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 um FlinkAPI. Você pode aprender mais sobre o Flink APIs na documentação do Apache Flink.

O Flink oferece quatro níveis de API abstração: Flink SQLAPI, DataStream API Tabela e Função de Processo, que são usados em conjunto com o. DataStream API Tudo isso é compatível com o Amazon Managed Service para Apache Flink. É aconselhável começar com um nível mais alto de abstração sempre que possível. No entanto, alguns recursos do Flink só estão disponíveis com o Datastream, API onde você pode criar seu aplicativo em Java, Python ou Scala. Você deve considerar o uso do Datastream seAPI:

- 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 cronômetros personalizados (por exemplo, para implementar janelas personalizadas ou tratamento tardio de eventos)

- Você quer poder modificar o fluxo do seu aplicativo sem redefinir o estado



Note

Escolhendo um idioma com o `DataStream`API:

- SQLpode ser incorporado em qualquer aplicativo Flink, independentemente da linguagem de programação escolhida.
- Se você planeja usar o DataStream API, nem todos os conectores são compatíveis com Python.
- Se você precisar de baixa latência/alto rendimento, considere o Java/Scala, independentemente do. API
- Se você planeja usar o Async IO nas Funções do Processo, API precisará usar Java.

A escolha do também API pode afetar sua capacidade de desenvolver a lógica do aplicativo sem precisar redefinir o estado. Isso depende de um recurso específico, a capacidade de definir UID operadores, que só está disponível no `DataStream` API para Java e Python. Para obter mais informações, consulte Definir UUIDs para todos os operadores na documentação do Apache Flink.

Comece a usar aplicativos de streaming de dados

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 sua IDE preferência 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:

- Serviço gerenciado para Apache Flink: como funciona
- Comece a usar o 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 padrões, SQL Python e Scala. Com alguns cliques no AWS Management Console, você pode iniciar um notebook sem servidor para consultar fluxos de dados e obter resultados em segundos. Para começar, recomendamos que você leia as seguintes seções:

- Use um notebook Studio com serviço gerenciado para Apache Flink
- Crie um caderno Studio

Serviço gerenciado para Apache Flink: como funciona

O Managed Service for Apache Flink é um serviço totalmente gerenciado da Amazon que permite usar um aplicativo Apache Flink para processar dados de streaming. Primeiro, você programa seu aplicativo Apache Flink e, em seguida, cria seu serviço gerenciado para o aplicativo Apache Flink.

Programe 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 tabela DataStream APIou a tabela API. Os outros Apache Flink também APIs estão disponíveis para você usar, mas são menos usados na criação de aplicativos de streaming.

As características dos dois APIs são as seguintes:

DataStream API

O modelo de DataStream API programação 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 o DataStream API fazem o seguinte:

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

Os aplicativos que usam o DataStream API podem ser escritos em Java ou Scala e podem ser lidos de um stream de dados do Kinesis, de um tópico da MSK Amazon 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 o DataStream API, consulte DataStream APIComponentes de revisão.

Tabela API

O modelo de API programação do Apache Flink Table 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 SQL tabela ou exibição.
- Fonte da tabela: usada para ler dados de uma fonte externa, como um MSK tópico da Amazon.
- Função de tabela: uma SQL consulta ou API chamada 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 Tabela API fazem o seguinte:

- Criam um `TableEnvironment` conectando-se a um `Table Source`.
- Crie uma tabela `TableEnvironment` usando SQL consultas ou API funções de tabela.
- Execute uma consulta na tabela usando Tabela API ou SQL
- Aplique transformações nos resultados da consulta usando funções de tabela ou SQL consultas.
- Gravam os resultados da consulta ou função em um `Table Sink`.

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

Para obter mais informações sobre como usar a tabelaAPI, consulteAPIComponentes da tabela de revisão.

Crie seu serviço gerenciado para o aplicativo Apache Flink

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

- Use propriedades de tempo de execução no Managed Service para Apache Flink: parâmetros que você pode fornecer ao seu aplicativo. Você pode alterar esses parâmetros sem recompilar o código do aplicativo.
- Implemente a tolerância a falhas no Managed Service for Apache Flink: como seu aplicativo se recupera de interrupções e reinicializações.
- Registro e monitoramento no Amazon Managed Service para Apache Flink: como seu aplicativo registra eventos no CloudWatch Logs.
- Implemente o escalonamento de aplicativos no Managed Service para Apache Flink: 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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Crie um serviço gerenciado para o aplicativo Apache Flink

Este tópico contém informações sobre a criação de um serviço gerenciado para o aplicativo Apache Flink.

Este tópico contém as seguintes seções:

- Crie seu serviço gerenciado para o código do aplicativo Apache Flink
- Crie seu serviço gerenciado para o aplicativo Apache Flink
- Inicie seu serviço gerenciado para o aplicativo Apache Flink
- Verifique seu serviço gerenciado para o aplicativo Apache Flink
- Habilite reversões do sistema para seu aplicativo Managed Service for Apache Flink

Crie seu serviço gerenciado para o código do aplicativo 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. Para obter informações sobre a atualização de aplicativos Managed Service for Apache Flink, consulte Use atualizações de versão in-loco para o Apache Flink.

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 para Apache Flink suporta JAR arquivos de até 512 MB de tamanho. Se você usar um JAR arquivo maior do que isso, seu aplicativo falhará ao iniciar.

Agora, os aplicativos podem usar o Java API de qualquer versão do Scala. Você deve agrupar a biblioteca padrão Scala de sua escolha em seus aplicativos Scala.

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

Especifique 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 `-Dflink.version` parâmetro. Por exemplo, se você estiver usando o Apache Flink 1.19.1, forneça o seguinte:

```
mvn package -Dflink.version=1.19.1
```

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

Crie seu serviço gerenciado para o aplicativo Apache Flink

Depois de criar o código do seu aplicativo, faça o seguinte para criar seu serviço gerenciado para o aplicativo 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 o upload do código do seu aplicativo,

consulte o Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink tutorial.

- 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 serviço gerenciado para o aplicativo Apache Flink usando o AWS console: Você pode criar e configurar seu aplicativo usando o AWS console.

    Quando você cria seu aplicativo usando o console, os recursos dependentes do seu aplicativo (como fluxos de CloudWatch registros, IAM funções e IAM políticas) 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 ver um tutorial sobre como usar o console para criar um aplicativo, consulte o Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink tutorial.
  - Crie seu serviço gerenciado para o aplicativo Apache Flink usando o AWS CLI: Você pode criar e configurar seu aplicativo usando o. AWS CLI

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

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

> **Note**
>
> Você pode alterar o `RuntimeEnvironment` de um aplicativo existente. Para saber como, consulte Use atualizações de versão in-loco para o Apache Flink.

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

Verifique seu serviço gerenciado para o aplicativo 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 no Amazon Managed Service para Apache Flink
- 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 streams do Kinesis, streams Firehose 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.

Habilite reversões do sistema para seu aplicativo Managed Service for Apache Flink

Com a capacidade de reversão do sistema, você pode obter maior disponibilidade do seu aplicativo Apache Flink em execução no Amazon Managed Service para Apache Flink. A opção por essa configuração permite que o serviço reverta automaticamente o aplicativo para a versão em execução anterior quando uma ação, como `UpdateApplication` ou `autoscaling` ocorre, falha no código ou na configuração.

> **Note**
>
> Para usar o recurso de reversão do sistema, você deve se inscrever atualizando seu aplicativo. Os aplicativos existentes não usarão automaticamente a reversão do sistema por padrão.

Como funciona

Quando você inicia uma operação de aplicativo, como uma ação de atualização ou escalabilidade, o Amazon Managed Service para Apache Flink primeiro tenta executar essa operação. Se detectar problemas que impedem o sucesso da operação, como erros de código ou permissões insuficientes, o serviço iniciará automaticamente uma operação. `RollbackApplication`

A reversão tenta restaurar o aplicativo para a versão anterior que foi executada com êxito, junto com o estado do aplicativo associado. Se a reversão for bem-sucedida, seu aplicativo continuará processando dados com o mínimo de tempo de inatividade usando a versão anterior. Se a reversão automática também falhar, o Amazon Managed Service para Apache Flink fará a transição do aplicativo para o `READY` status, para que você possa realizar outras ações, incluindo corrigir o erro e repetir a operação.

Você deve optar por usar as reversões automáticas do sistema. Você pode habilitá-lo usando o console ou API para todas as operações em seu aplicativo a partir de agora.

O exemplo a seguir de solicitação para a `UpdateApplication` ação permite reversões do sistema para um aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "ApplicationSystemRollbackConfigurationUpdate": {
         "RollbackEnabledUpdate": "true"
       }
    }
}
```

Analise cenários comuns para reversão automática do sistema

Os cenários a seguir ilustram onde as reversões automáticas do sistema são benéficas:

- Atualizações do aplicativo: se você atualizar seu aplicativo com um novo código que contém bugs ao inicializar a tarefa do Flink por meio do método principal, a reversão automática permite que a versão de trabalho anterior seja restaurada. Outros cenários de atualização em que as reversões do sistema são úteis incluem:
  - Se seu aplicativo for atualizado para ser executado com um paralelismo maior que. maxParallelism
  - Se seu aplicativo for atualizado para ser executado com sub-redes incorretas para um VPC aplicativo, isso resultará em uma falha durante a inicialização da tarefa do Flink.
- Atualizações da versão do Flink: quando você atualiza para uma nova versão do Apache Flink e o aplicativo atualizado encontra um problema de compatibilidade de instantâneos, a reversão do sistema permite que você reverta automaticamente para a versão anterior do Flink.
- AutoScaling: quando o aplicativo se expande, mas apresenta problemas de restauração a partir de um ponto de salvamento, devido à incompatibilidade do operador entre o instantâneo e o gráfico de tarefas do Flink.

Use a operação APIs para reversões do sistema

Para fornecer melhor visibilidade, o Amazon Managed Service para Apache Flink tem duas operações APIs relacionadas a aplicativos que podem ajudá-lo a rastrear falhas e reversões de sistema relacionadas.

`ListApplicationOperations`

Isso API lista todas as operações realizadas no aplicativo, incluindo,`UpdateApplication`, `MaintenanceRollbackApplication`, e outras em ordem cronológica inversa. O exemplo a seguir de solicitação para a `ListApplicationOperations` ação lista as 10 primeiras operações do aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "Limit": 10
}
```

Este exemplo de solicitação a seguir `ListApplicationOperations` ajuda a filtrar a lista para atualizações anteriores no aplicativo:

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

```
   "operation": "UpdateApplication"
}
```

`DescribeApplicationOperation`

Isso API fornece informações detalhadas sobre uma operação específica listada por`ListApplicationOperations`, incluindo o motivo da falha, se aplicável. O exemplo a seguir de solicitação para a `DescribeApplicationOperation` ação lista os detalhes de uma operação específica do aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "OperationId": "xyzoperation"
}
```

Para obter informações sobre a solução de problemas, consulte Melhores práticas de reversão do sistema.

Execute um serviço gerenciado para o aplicativo 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 trabalhos do Apache Flink, consulte Trabalhos e agendamento na documentação do Apache Flink.

Identifique a candidatura e o 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 READY status até que você o execute.
  - RUNNING: um aplicativo que foi iniciado com sucesso está no RUNNING status.
- 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 StopApplicationação com o `Force` parâmetro definido como. `true` Esses status incluem o seguinte:
  - `STARTING`: Ocorre após a StartApplicationação. O aplicativo está passando do status `READY` para `RUNNING`.
  - `STOPPING`: Ocorre após a StopApplicationação. O aplicativo está passando do status `RUNNING` para `READY`.
  - `DELETING`: Ocorre após a DeleteApplicationação. O aplicativo está em processo de ser excluído.
  - `UPDATING`: Ocorre após a UpdateApplicationação. O aplicativo está sendo atualizado e voltará ao status `RUNNING` ou `READY`.
  - `AUTOSCALING`: O aplicativo tem a `AutoScalingEnabled` propriedade de ParallelismConfigurationdefinir como`true`, e o serviço está aumentando o paralelismo do aplicativo. Quando o aplicativo está nesse status, a única API ação válida que você pode usar é a StopApplicationação com o `Force` parâmetro definido como`true`. Para obter mais informações sobre escalabilidade, consulte Use o escalonamento automático no Managed Service para Apache Flink.
  - `FORCE_STOPPING`: Ocorre depois que a StopApplicationação 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 RollbackApplicationação 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`.
  - `MAINTENANCE`: Ocorre enquanto o Managed Service for Apache Flink aplica patches ao seu aplicativo. Para obter mais informações, consulte Gerencie tarefas de manutenção do Managed Service for Apache Flink.

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

- 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 CloudWatch registro de sua inscrição em busca de alterações de status.

Execute cargas de trabalho em lote

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

Analise os recursos do aplicativo Managed Service for Apache Flink

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.

Serviço gerenciado para recursos do aplicativo Apache Flink

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

Um KPU representa os seguintes recursos do sistema:

- Um CPU núcleo

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

KPUsexecute 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 `Parallelism` configuração do aplicativo, dividido pela `ParallelismPerKPU` configuração do aplicativo.

Para obter mais informações sobre paralelismo de aplicativo, consulte Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

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 um único. KPU O número de slots atribuídos a um único KPU é igual à `ParallelismPerKPU` configuração do seu aplicativo. Para obter mais informações sobre slots de tarefas, consulte Job Scheduling na documentação do Apache Flink.

Paralelismo de operadores

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 operadores, consulte Operadores 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.

DataStream APIComponentes de revisão

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

Esta seção descreve os diferentes componentes que movem, transformam e rastreiam dados:

- Use conectores para mover dados no Managed Service for Apache Flink com o DataStream API: esses componentes movem dados entre seu aplicativo e fontes de dados e destinos externos.
- Transforme dados usando operadores no Managed Service for Apache Flink com o DataStream API: esses componentes transformam ou agrupam elementos de dados em seu aplicativo.
- Rastreie eventos no Managed Service for Apache Flink usando o DataStream API: Este tópico descreve como o Managed Service for Apache Flink rastreia eventos ao usar o. DataStream API

Use conectores para mover dados no Managed Service for Apache Flink com o DataStream API

No Amazon Managed Service para Apache Flink DataStream API, 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:

- Adicione fontes de dados de streaming ao Managed Service for Apache Flink: forneça dados para seu aplicativo a partir de um fluxo de dados do Kinesis, arquivo ou de outra fonte de dados.
- Grave dados usando coletores no Managed Service for Apache Flink: envie dados do seu aplicativo para um stream de dados do Kinesis, stream Firehose ou outro destino de dados.
- Use E/S assíncrona no serviço gerenciado para Apache Flink: fornece acesso assíncrono a uma fonte de dados (como um banco de dados) para enriquecer os eventos de fluxo.

Conectores disponíveis

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 executá-los nas regiões do Oriente Médio (), Ásia-Pacífico (HyderabadUAE), Israel (Tel Aviv), Europa (Zurique), Oriente Médio (), Ásia-Pacífico (UAEMelbourne) ou Ásia-Pacífico (Jacarta), talvez seja necessário reconstruir seu arquivo de aplicativos com um conector atualizado ou atualizar para o Flink 1.18.
Os conectores Apache Flink são armazenados em seus próprios repositórios de código aberto. Se você estiver atualizando para a versão 1.18 ou posterior, deverá atualizar suas dependências. Para acessar o repositório dos AWS conectores Apache Flink, consulte. flink-connector-aws
A seguir estão as diretrizes recomendadas:

Atualizações de conectores

\ c F	Conector usado	Resolução
1	EFO	Ao fazer o upgrade para o Amazon Managed Service for Apache Flink versão 1.15, verifique se você está usando o conector mais

V d F	Conector usado	Resolução
		recente. EFO Essa deve ser qualquer versão 1.15.3 ou posterior. Para obter mais informações, consulte: FLINK-29324.
1	Coletor Amazon Data Firehose	Ao fazer o upgrade para o Amazon Managed Service for Apache Flink versão 1.15, verifique se você está usando o Amazon Data Firehose Sink mais recente. Coletor Amazon Data Firehose
1	Conectores Kafka	Ao fazer o upgrade para o Amazon Managed Service for Apache Flink versão 1.15, verifique se você está usando o conector Kafka mais recente. APIs O Apache Flink foi descontinuado e. FlinkKafkaConsumerFlinkKafkaProducer Esses APIs para o coletor Kafka não podem se comprometer com o Kafka para o Flink 1.15. Certifique-se de que você está usando KafkaSourceKafkaSinke.

V c F	Conector usado	Resolução
1 - 1	Firehose	Seu aplicativo depende de uma versão desatualizada do conector Firehose que não conhece as regiões mais AWS recentes. Reconstrua o arquivo do seu aplicativo com o conector Firehose versão 2.1.0. v2.1.0
1	Kinesis	Seu aplicativo depende de uma versão desatualizada do conector Flink Kinesis que não conhece as regiões mais recentes. AWS Reconstrua o arquivo do seu aplicativo com o conector Flink Kinesis versão 1.6.1. https://github.com/awslabs/amazon-kinesis-connector-flink/árvore/1.6.1

V d F	Conector usado	Resolução
1	Kinesis	Seu aplicativo depende de uma versão desatualizada do conector Flink Kinesis que não conhece as regiões mais recentes. AWS Reconstrua o arquivo do seu aplicativo com o conector Flink Kinesis versão 2.4.1. https://github.com/awslabs/amazon-kinesis-connector-flink/árvore/2.4.1
1 e 1	Kinesis	Seu aplicativo depende de uma versão desatualizada do conector Flink Kinesis que não conhece as regiões mais recentes. AWS Infelizmente, o Flink não lança mais patches ou correções de 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.

Adicione fontes de dados de streaming 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 disponíveis para os serviços da Amazon

Use streams de dados do Kinesis

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

Criar uma **`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 Baixe e examine o código Java de streaming do Apache Flink.

Crie um **`FlinkKinesisConsumer`** que use um EFO consumidor

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

Se um consumidor do Kinesis usaEFO, 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 EFO com o consumidor Kinesis, consulte FLIP-128: Enhanced Fan Out for AWS Kinesis Consumers.

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

- RECORD_ PUBLISHER _TYPE: defina esse parâmetro EFOpara que seu aplicativo use um EFO consumidor para acessar os dados do Kinesis Data Stream.

- EFO_ CONSUMER _NAME: defina esse parâmetro como um valor de string 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 usarEFO, 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 serviço gerenciado para o aplicativo Apache Flink que usa um EFO consumidor, consulte. Consumidor EFO

Use a Amazon MSK

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

Criar 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.

Grave dados usando coletores no Managed Service for Apache Flink

No código do seu aplicativo, você pode usar qualquer conector de coletor do Apache Flink para gravar em sistemas externos, incluindo AWS serviços, como Kinesis Data Streams e DynamoDB.

O Apache Flink também fornece coletores para arquivos e soquetes, e você pode implementar coletores personalizados. Entre os vários coletores suportados, os seguintes são usados com frequência:

Use streams de dados do Kinesis

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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Use o Apache Kafka e o Amazon Managed Streaming para Apache Kafka () MSK

O conector Apache Flink Kafka fornece amplo suporte para publicação de dados no Apache Kafka e na Amazon, incluindo garantias de uma única vez. MSK Para aprender a escrever no Kafka, consulte exemplos de conectores Kafka na documentação do Apache Flink.

Use o 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”.

Use Firehose

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

Tópicos

- Criar uma FlinkKinesisFirehoseProducer
- Exemplo de código FlinkKinesisFirehoseProducer

Criar uma **`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 stream de dados do Apache Flink para o serviço 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 obter um tutorial completo sobre como usar o coletor Firehose, consulte. the section called “Pia Firehose”

Use E/S assíncrona no serviço gerenciado para 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.

Transforme dados usando operadores no Managed Service for Apache Flink com o DataStream API

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:

- Use operadores de transformação
- Use operadores de agregação

Use operadores de transformação

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

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");
        }
    }
);
```

Use 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 mais exemplos de código, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

Rastreie eventos no Managed Service for Apache Flink usando o DataStream API

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 streaming usando`setStreamTimeCharacteristic`.

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

Para obter mais informações sobre timestamps, consulte Gerando marcas d'água na documentação do Apache Flink.

APIComponentes da tabela de revisão

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

Esta seção contém os seguintes tópicos:

- Use API conectores de mesa: esses componentes movem dados entre seu aplicativo e fontes de dados e destinos externos.
- Revise os atributos de API tempo da tabela: Este tópico descreve como o Managed Service for Apache Flink rastreia eventos ao usar a Tabela. API

Use API conectores de mesa

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. AWS

Com a tabela Apache FlinkAPI, você pode usar os seguintes tipos de conectores:

- APIFontes da tabela: você usa conectores API de origem de tabela para criar tabelas dentro de você `TableEnvironment` usando API chamadas ou SQL consultas.
- APIPias de mesa: Você usa SQL comandos para gravar dados de tabela em fontes externas, como um MSK tópico da Amazon ou um bucket do Amazon S3.

APIFontes da 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 MSK tópico da Amazon:

```
//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 SQL conectores na documentação do Apache Flink.

APIPias de mesa

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

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 Conectores compatíveis 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 AWS serviços, como Amazon e Amazon MSK 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.

Revise os atributos de API tempo da 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 tabela Apache Flink API cria janelas com base em tempos recordes, você define quais desses carimbos de data/hora ela usa usando o método. `setStreamTimeCharacteristic`

Para obter mais informações sobre o uso de timestamps com a tabelaAPI, consulte Time Attributes and Timely Stream Processing na documentação do Apache Flink.

Use Python com serviço gerenciado para Apache Flink

> **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.

A versão 1.18.1 do Apache Flink inclui suporte para criar aplicativos usando a versão 3.10 do Python. Para obter mais informações, consulte Flink Python Docs. 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 tabela Python API é um invólucro em torno da tabela Java. API Para obter informações sobre a tabela PythonAPI, consulte o APItutorial da tabela na documentação do Apache Flink.

Programe seu serviço gerenciado para o aplicativo Apache Flink Python

Você codifica seu serviço gerenciado para o aplicativo Apache Flink para Python usando a tabela Python do Apache Flink. API O mecanismo Apache Flink traduz instruções de API tabela Python (em execução na VM Python) em instruções de API tabela Java (em execução na VM Java).

Você usa a tabela Python API fazendo o seguinte:

- 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 tabela Python API no Managed Service para Apache Flink, consulte. Comece a usar o Amazon Managed Service para Apache Flink para Python

Leia e grave dados de streaming

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

Criar uma tabela

O exemplo de código a seguir demonstra uma função definida pelo usuário que cria uma SQL consulta. A SQL consulta 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)
```

Leia dados de streaming

O exemplo de código a seguir demonstra como usar a `CreateTable` SQL consulta 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))
```

Grave dados de streaming

O exemplo de código a seguir demonstra como usar a SQL consulta do `CreateTable` exemplo para criar uma referência de tabela de saída e como usar a para interagir com as tabelas `StatementSet` para gravar 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))
```

Leia as propriedades do tempo de execução

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:

- Usando a CreateApplicationação.
- Usando a UpdateApplicationação.
- 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]
```

Crie 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 JAR arquivos

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

Crie seu serviço gerenciado para o aplicativo Apache Flink Python

Especifique 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, você cria seu aplicativo usando o console ou a CreateApplicationação.

Ao criar seu aplicativo usando a CreateApplicationação, 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 JAR arquivo Java 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 Interface de 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"
          }
        },
```

Monitore seu serviço gerenciado para o aplicativo Apache Flink Python

Você usa o CloudWatch log do seu aplicativo para monitorar seu serviço gerenciado para o aplicativo 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.

Registros de consulta com o CloudWatch Insights

A consulta do CloudWatch Insights a seguir pesquisa registros 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
```

Use propriedades de tempo de execução no Managed Service para Apache Flink

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

Este tópico contém as seguintes seções:

- Gerencie propriedades de tempo de execução usando o console
- Gerencie propriedades de tempo de execução usando o CLI
- Acesse propriedades de tempo de execução em um aplicativo Managed Service for Apache Flink

Gerencie propriedades de tempo de execução usando o console

Você pode adicionar, atualizar ou remover propriedades de tempo de execução do seu aplicativo Managed Service for Apache Flink usando o. AWS Management Console

> **Note**
>
> Se você estiver usando uma versão anterior compatível do Apache Flink e quiser atualizar seus aplicativos existentes para o Apache Flink 1.19.1, você pode fazer isso usando atualizações de versão do Apache Flink in-loco. Com as atualizações de versão no local, você mantém a rastreabilidade do aplicativo ARN em relação a uma única versão do Apache Flink, incluindo instantâneos, registros, métricas, tags, configurações do Flink e muito mais. Você pode usar esse recurso em `RUNNING` um `READY` estado. Para obter mais informações, consulte Use atualizações de versão in-loco para o Apache Flink.

Atualizar propriedades de runtime for Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
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.

Gerencie propriedades de tempo de execução usando o CLI

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

Esta seção inclui exemplos de solicitações de API ações para configurar propriedades de tempo de execução para um aplicativo. Para obter informações sobre como usar um JSON arquivo como entrada para uma API ação, consulteCódigo de exemplo do Managed Service for Apache Flink API.

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

Adicione propriedades de tempo de execução 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_19",
    "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 tempo de execução 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 tempo de execução

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.

Acesse propriedades de tempo de execução 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 exemplos de código, consulte Exemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

Use conectores Apache Flink com o Managed Service para Apache Flink

Os conectores Apache Flink são componentes de software que movem dados para dentro e para fora de um aplicativo Amazon Managed Service para 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:

- Fontes: forneça dados para seu aplicativo a partir de um stream de dados, arquivo, tópico do Apache Kafka, arquivo ou outras fontes de dados do Kinesis.
- Coletores: envie dados do seu aplicativo para um stream de dados do Kinesis, um stream do Firehose, um tópico do Apache Kafka ou outros destinos de dados.

- E/S assíncrona: fornece acesso assíncrono a uma fonte de dados, como um banco de dados, para enriquecer os fluxos.

Os conectores Apache Flink são armazenados em seus próprios repositórios de origem. A versão e o artefato dos conectores Apache Flink mudam dependendo da versão do Apache Flink que você está usando e se você está usando a Tabela ou. DataStream SQL API

O Amazon Managed Service para Apache Flink oferece suporte a mais de 40 conectores de origem e coletor pré-construídos do Apache Flink. A tabela a seguir fornece um resumo dos conectores mais populares e suas versões associadas. Você também pode criar coletores personalizados usando a estrutura Async-sink. Para obter mais informações, consulte The Generic Asynchronous Base Sink na documentação do Apache Flink.

Para acessar o repositório dos AWS conectores Apache Flink, consulte. flink-connector-aws

Conectores para versões Flink

Conector	Flink versão 1.15	Flink versão 1.18	Flink versão 1.19
Kinesis Data Stream — DataStream Fonte e tabela API	flink-connector-kinesis, 1.15.4	flink-connector-kinesis, 4.3.0-1.18	flink-connector-kinesis, 4.3.0-1.19
Kinesis Data Stream — coletor — DataStream e tabela API	flink-connector-aws-kinesis-streams, 1.15.4	flink-connector-aws-kinesis-streams, 4.3.0-1.18	flink-connector-aws-kinesis-streams, 4.3.0-1.19
Kinesis Data Streams — Fonte/coletor — SQL	flink-sql-connector-kinesis, 1.15.4	flink-sql-connector-kinesis, 4.3.0-1.18	flink-sql-connector-kinesis, 4.3.0-1.19
Kafka - DataStream e mesa API	flink-connector-kafka, 1.15.4	flink-connector-kafka, 3.2.0-1.18	flink-connector-kafka, 3.2.0-1.19
Kafka - SQL	flink-sql-connector-kafka, 1.15.4	flink-sql-connector-kafka, 3.2.0-1.18	flink-sql-connector-kafka, 3.2.0-1.19

Conector	Flink versão 1.15	Flink versão 1.18	Flink versão 1.19
Firehose - DataStream e mesa API	flink-connector-aws-kinesis- mangueira de incêndio, 1.15.4	flink-connector-aws-firehose, 4.3.0-1.18	flink-connector-aws-firehose, 4.3.0-1.19
Firehose - SQL	flink-sql-connector-aws-kinesis-fire mangueira, 1.15.4	flink-sql-connector-aws-mangueira de incêndio, 4.3.0-1.18	flink-sql-connector-aws-mangueira de incêndio, 4.3.0-1.19
DynamoDB DataStream — e tabela API	flink-connector-dynamodb, 3.0.0-1.15	flink-connector-dynamodb, 4.3.0-1.18	flink-connector-dynamodb, 4.3.0-1.19
DynamoDB - SQL	flink-sql-connector-dynamodb, 3.0.0-1.15	flink-sql-connector-dynamodb, 4.3.0-1.18	flink-sql-connector-dynamodb, 4.3.0-1.19
OpenSearch - DataStream e mesa API	-	flink-connector-opensearch, 1.2.0-1.18	flink-connector-opensearch, 1.2.0-1.19
OpenSearch - SQL	-	flink-sql-connector-opensearch, 1.2.0-1.18	flink-sql-connector-opensearch, 1.2.0-1.19

Para saber mais sobre conectores no Amazon Managed Service para Apache Flink, consulte:

- DataStream APIconectores
- APIConectores de mesa

Implemente a 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 up-to-date backup 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 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 Gerencie backups de aplicativos usando instantâneos.

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

Configurar o ponto de verificação no Managed Service para 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 `UpdateApplication`APIoperações `CreateApplication`ou:

- `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: verdadeiro

- CheckpointInterval: 60000
- MinPauseBetweenCheckpoints: 5000

Se ConfigurationTypeestiver definido comoDEFAULT, os valores anteriores serão usados, mesmo que sejam definidos para outros valores usando o 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`.

Analise exemplos de pontos de verificação API

Esta seção inclui exemplos de solicitações de API ações para configurar o ponto de verificação para um aplicativo. Para obter informações sobre como usar um JSON arquivo como entrada para uma API ação, consulteCódigo de exemplo do Managed Service for Apache Flink API.

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_19",
   "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_19",
   "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
         }
      }
   }
}
```

Gerencie backups de aplicativos usando instantâneos

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 Savepoints do Apache Flink, consulte Savepoints na documentação do Apache Flink. Usando instantâneos, 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 instantâneos 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 instantâneos usando as seguintes ações: API

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

Para saber o limite do número de snapshots por aplicativo, consulte Serviço gerenciado para cota de notebooks Apache Flink e Studio. 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:

- Analise exemplos de instantâneos API

Gerencie a criação automática de instantâneos

Se `SnapshotsEnabled` estiver definido como `true` no ApplicationSnapshotConfigurationpara o aplicativo, o Managed Service for Apache Flink cria e usa automaticamente instantâneos 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 instantâneo é gerenciado pelo serviço, mas você pode ver o instantâneo usando a ListApplicationSnapshotsação. 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 instantâneos usando a DeleteApplicationSnapshotação antes de criar mais instantâneos manualmente.

Restauração 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 `AllowNonRestoredState` parâmetro do FlinkRunConfigurationpara `true` usar a UpdateApplicationação.

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 instantâneos, consulte Savepoints 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 two-phase-commit coletores, 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 snapshot definindo o parâmetro da `ApplicationRestoreType` ação como. StartApplication`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.

Analise exemplos de instantâneos API

Esta seção inclui exemplos de solicitações de API ações para o uso de instantâneos com um aplicativo. Para obter informações sobre como usar um JSON arquivo como entrada para uma API ação, consulteCódigo de exemplo do Managed Service for Apache Flink API.

Habilitar instantâneos 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 instantâneos 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 instantâneo do 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,
}
```

Reinicie 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"
      }
   }
}
```

Reinicie 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"
      }
   }
}
```

Reinicie um aplicativo sem tirar um instantâneo

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"
      }
   }
}
```

Use atualizações de versão in-loco para o Apache Flink

Com atualizações de versão in-loco para o Apache Flink, você mantém a rastreabilidade do aplicativo em relação a uma única versão do Apache Flink. ARN Isso inclui instantâneos, registros, métricas, tags, configurações do Flink, aumentos no limite de recursos e muito mais. VPCs

Você pode realizar atualizações de versão in-loco para o Apache Flink para atualizar aplicativos existentes para uma nova versão do Flink no Amazon Managed Service para Apache Flink. Para executar essa tarefa, você pode usar o AWS CLI AWS CloudFormation AWS SDK,, ou AWS Management Console o.

> **Note**
>
> Você não pode usar atualizações de versão in-loco para o Apache Flink com o Amazon Managed Service para Apache Flink Studio.

Este tópico contém as seguintes seções:

Atualize aplicativos usando atualizações de versão in-loco para o Apache Flink

Antes de começar, recomendamos que você assista a este vídeo: Atualizações de versão in-loco.

Para realizar atualizações de versão in-loco para o Apache Flink, você pode usar o AWS CLI,, AWS CloudFormation, AWS SDK ou o. AWS Management Console Você pode usar esse recurso com qualquer aplicativo existente que você usa com o Managed Service for Apache Flink em um estado `READY` ou`RUNNING`. Ele usa o UpdateApplication API para adicionar a capacidade de alterar o tempo de execução do Flink.

Antes da atualização: atualize seu aplicativo Apache Flink

Ao escrever seus aplicativos Flink, você os agrupa com suas dependências em um aplicativo JAR e os carrega em seu bucket do Amazon JAR S3. A partir daí, o Amazon Managed Service para Apache Flink executa o trabalho no novo tempo de execução do Flink que você selecionou. Talvez seja necessário atualizar seus aplicativos para obter compatibilidade com o tempo de execução do Flink para o qual você deseja fazer o upgrade. Pode haver inconsistências entre as versões do Flink que fazem com que a atualização da versão falhe. Geralmente, isso ocorre com conectores para fontes

(entrada) ou destinos (coletores, saída) e dependências do Scala. O Flink 1.15 e versões posteriores no Managed Service for Apache Flink são independentes de Scala, e você JAR deve conter a versão do Scala que você planeja usar.

Para atualizar seu aplicativo

1. Leia os conselhos da comunidade Flink sobre como atualizar aplicativos com o estado. Consulte Atualização de aplicativos e versões do Flink.
2. Leia a lista de problemas e limitações conhecidos. Consulte Precauções e problemas conhecidos com atualizações de aplicativos.
3. Atualize suas dependências e teste seus aplicativos localmente. Essas dependências normalmente são:
    1. O tempo de execução do Flink e. API
    2. Conectores recomendados para o novo tempo de execução do Flink. Você pode encontrá-los nas versões Release para o tempo de execução específico para o qual deseja atualizar.
    3. Scala — O Apache Flink é independente de Scala, começando com e incluindo o Flink 1.15. Você deve incluir as dependências do Scala que deseja usar em seu aplicativo. JAR
4. Crie um novo aplicativo JAR no arquivo zip e faça o upload para o Amazon S3. Recomendamos que você use um nome diferente do JAR /zipfile. Se precisar reverter, você usará essas informações.
5. Se você estiver executando aplicativos com estado, é altamente recomendável que você tire um instantâneo do seu aplicativo atual. Isso permite reverter de forma contínua se você encontrar problemas durante ou após a atualização.

Atualize seu aplicativo para uma nova versão do Apache Flink

Você pode atualizar seu aplicativo Flink usando a UpdateApplicationação.

Você pode chamar o `UpdateApplication` API de várias maneiras:

- Use o fluxo de trabalho de configuração existente no AWS Management Console.
    - Acesse a página do seu aplicativo no AWS Management Console.
    - Selecione Configurar.
    - Selecione o novo tempo de execução e o snapshot a partir do qual você deseja começar, também conhecido como configuração de restauração. Use a configuração mais recente como

configuração de restauração para iniciar o aplicativo a partir do instantâneo mais recente. Aponte para o novo aplicativo atualizado JAR /zip no Amazon S3.

- Use a ação AWS CLI de atualização do aplicativo.
- Uso AWS CloudFormation (CFN).
  - Atualize o RuntimeEnvironmentcampo. Anteriormente, AWS CloudFormation excluía o aplicativo e criava um novo, fazendo com que seus instantâneos e outros históricos do aplicativo fossem perdidos. Agora AWS CloudFormation atualiza seu RuntimeEnvironment local e não exclui seu aplicativo.
- Use AWS SDK o.
  - Consulte a SDK documentação da linguagem de programação de sua escolha. Veja UpdateApplication.

Você pode realizar a atualização enquanto o aplicativo estiver no `RUNNING` estado ou enquanto o aplicativo estiver parado no `READY` estado. O Amazon Managed Service para Apache Flink valida para verificar a compatibilidade entre a versão original do tempo de execução e a versão do tempo de execução de destino. Essa verificação de compatibilidade é executada quando você executa UpdateApplicationenquanto está no `RUNNING` estado ou na próxima, StartApplicationse você atualiza enquanto está no `READY` estado.

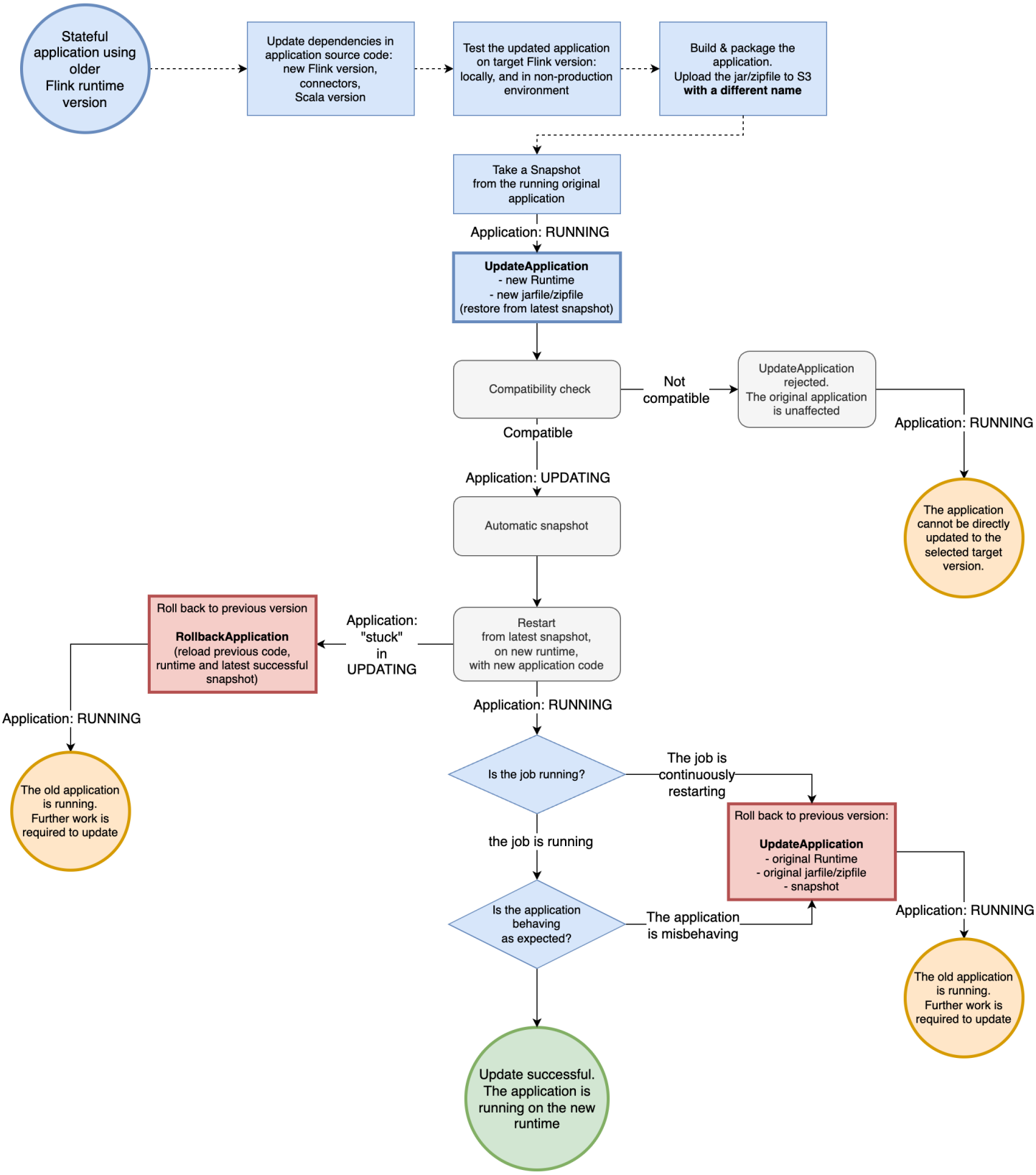
Atualizar um aplicativo no **`RUNNING`** estado

O exemplo a seguir mostra a atualização de um aplicativo no `RUNNING` estado chamado `UpgradeTest` Flink 1.18 no Leste dos EUA (Norte da Virgínia) usando AWS CLI e iniciando o aplicativo atualizado a partir do snapshot mais recente.

```
aws --region us-east-1 kinesisanalyticsv2 update-application \
--application-name UpgradeTest --runtime-environment-update "FLINK-1_18" \
--application-configuration-update '{"ApplicationCodeConfigurationUpdate": '\
'{"CodeContentUpdate": {"S3ContentLocationUpdate": '\
'{"FileKeyUpdate": "flink_1_18_app.jar"}}}}' \
 --run-configuration-update '{"ApplicationRestoreConfiguration": '\
 '{"ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"}}' \
 --current-application-version-id ${current_application_version}
```

- Se você habilitou os instantâneos do serviço e deseja continuar o aplicativo a partir do snapshot mais recente, o Amazon Managed Service para Apache Flink verifica se o tempo de execução do `RUNNING` aplicativo atual é compatível com o tempo de execução de destino selecionado.
- Se você especificou um snapshot a partir do qual continuar o tempo de execução de destino, o Amazon Managed Service para Apache Flink verifica se o tempo de execução de destino é compatível com o snapshot especificado. Se a verificação de compatibilidade falhar, sua solicitação de atualização será rejeitada e seu aplicativo permanecerá inalterado no `RUNNING` estado.
- Se você optar por iniciar seu aplicativo sem um snapshot, o Amazon Managed Service para Apache Flink não executará nenhuma verificação de compatibilidade.
- Se o aplicativo atualizado falhar ou ficar preso em um `UPDATING` estado transitivo, siga as instruções na Reverter atualizações de aplicativos seção para retornar ao estado íntegro.

Fluxo de processo para execução de aplicativos de estado


Stateful application using older Flink runtime version
Update dependencies in application source code: new Flink version, connectors, Scala version
Test the updated application on target Flink version: locally, and in non-production environment
Build & package the application. Upload the jar/zipfile to S3 with a different name
Take a Snapshot from the running original application
Application: RUNNING
UpdateApplication - new Runtime - new jarfile/zipfile (restore from latest snapshot)
Compatibility check
Not compatible
UpdateApplication rejected. The original application is unaffected
Application: RUNNING
The application cannot be directly updated to the selected target version.
Compatible
Application: UPDATING
Automatic snapshot
Restart from latest snapshot, on new runtime, with new application code
Application: "stuck" in UPDATING
Roll back to previous version
RollbackApplication (reload previous code, runtime and latest successful snapshot)
Application: RUNNING
The old application is running. Further work is required to update
Application: RUNNING
Is the job running?
The job is continuously restarting
Roll back to previous version:
UpdateApplication - original Runtime - original jarfile/zipfile - snapshot
Application: RUNNING
The old application is running. Further work is required to update
the job is running
Is the application behaving as expected?
The application is misbehaving
Update successful. The application is running on the new runtime

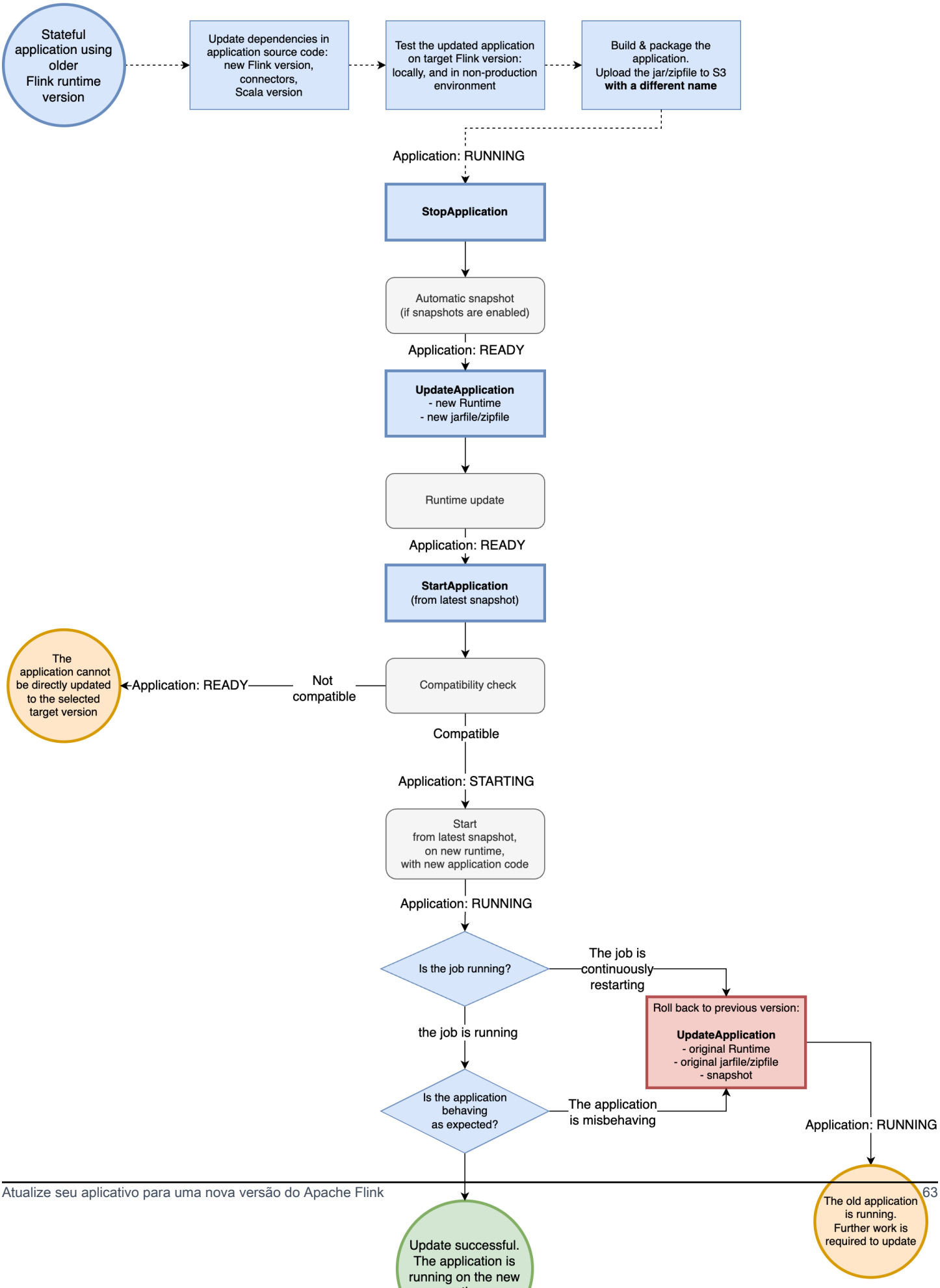
Atualizar um aplicativo no READYestado

O exemplo a seguir mostra a atualização de um aplicativo no `READY` estado chamado `UpgradeTest` Flink 1.18 no Leste dos EUA (Norte da Virgínia) usando o. AWS CLI Não há um instantâneo especificado para iniciar o aplicativo porque o aplicativo não está em execução. Você pode especificar um instantâneo ao emitir a solicitação inicial do aplicativo.

```
aws --region us-east-1 kinesisanalyticsv2 update-application \
--application-name UpgradeTest --runtime-environment-update "FLINK-1_18" \
--application-configuration-update '{"ApplicationCodeConfigurationUpdate": '\
'{"CodeContentUpdate": {"S3ContentLocationUpdate": '\
'{"FileKeyUpdate": "flink_1_18_app.jar"}}}}' \
 --current-application-version-id ${current_application_version}
```

- Você pode atualizar o tempo de execução de seus aplicativos em `READY` estado para qualquer versão do Flink. O Amazon Managed Service para Apache Flink não executa nenhuma verificação até você iniciar seu aplicativo.
- O Amazon Managed Service para Apache Flink só executa verificações de compatibilidade com o snapshot que você selecionou para iniciar o aplicativo. Essas são verificações básicas de compatibilidade de acordo com a Tabela de Compatibilidade do Flink. Eles verificam apenas a versão do Flink com a qual o instantâneo foi tirado e a versão do Flink que você está almejando. Se o tempo de execução do Flink do snapshot selecionado for incompatível com o novo tempo de execução do aplicativo, a solicitação inicial poderá ser rejeitada.

Fluxo de processo para aplicativos de estado pronto


Stateful application using older Flink runtime version
Update dependencies in application source code: new Flink version, connectors, Scala version
Test the updated application on target Flink version: locally, and in non-production environment
Build & package the application. Upload the jar/zipfile to S3 with a different name
Application: RUNNING
StopApplication
Automatic snapshot (if snapshots are enabled)
Application: READY
UpdateApplication - new Runtime - new jarfile/zipfile
Runtime update
Application: READY
StartApplication (from latest snapshot)
Compatibility check
Not compatible
Application: READY
The application cannot be directly updated to the selected target version
Compatible
Application: STARTING
Start from latest snapshot, on new runtime, with new application code
Application: RUNNING
Is the job running?
The job is continuously restarting
Roll back to previous version: UpdateApplication - original Runtime - original jarfile/zipfile - snapshot
the job is running
Is the application behaving as expected?
The application is misbehaving
Application: RUNNING
Update successful. The application is running on the new
The old application is running. Further work is required to update

Reverter atualizações de aplicativos

Se você tiver problemas com o aplicativo ou encontrar inconsistências no código do aplicativo entre as versões do Flink, poderá reverter usando o AWS CLI, AWS CloudFormation AWS SDK, ou o. AWS Management Console Os exemplos a seguir mostram como é a reversão em diferentes cenários de falha.

A atualização do tempo de execução foi bem-sucedida, o aplicativo está funcionando**RUNNING**, mas o trabalho está falhando e está sendo reiniciado continuamente

Suponha que você esteja tentando atualizar um aplicativo com estado chamado `TestApplication` de Flink 1.15 para Flink 1.18 no Leste dos EUA (Norte da Virgínia). No entanto, o aplicativo Flink 1.18 atualizado está falhando ao iniciar ou está sendo reiniciado constantemente, mesmo que o aplicativo esteja em estado. `RUNNING` Esse é um cenário de falha comum. Para evitar mais tempo de inatividade, recomendamos que você reverta seu aplicativo imediatamente para a versão anterior em execução (Flink 1.15) e diagnostique o problema posteriormente.

Para reverter o aplicativo para a versão anterior em execução, use o AWS CLI comando rollback-application ou a ação. RollbackApplicationAPI Essa API ação reverte as alterações que você fez que resultaram na versão mais recente. Em seguida, ele reinicia seu aplicativo usando o último snapshot bem-sucedido.

É altamente recomendável que você tire um instantâneo do seu aplicativo existente antes de tentar fazer o upgrade. Isso ajudará a evitar a perda de dados ou a necessidade de reprocessar dados.

Nesse cenário de falha, não AWS CloudFormation reverterá o aplicativo para você. Você deve atualizar o CloudFormation modelo para apontar para o tempo de execução anterior e para o código anterior para CloudFormation forçar a atualização do aplicativo. Caso contrário, CloudFormation presume que seu aplicativo tenha sido atualizado ao fazer a transição para o `RUNNING` estado.

Revertendo um aplicativo que está preso **UPDATING**

Se seu aplicativo ficar preso no `AUTOSCALING` estado `UPDATING` or após uma tentativa de atualização, o Amazon Managed Service para Apache Flink oferece o AWS CLI comando rollback-applications, ou a RollbackApplicationsAPIação que pode reverter o aplicativo para a versão anterior ao travamento ou estado bloqueado. `UPDATING` `AUTOSCALING` Isso API reverte as alterações que você fez que causaram a paralisação `UPDATING` ou o estado `AUTOSCALING` transitivo do aplicativo.

Práticas recomendadas e recomendações gerais para atualizações de aplicativos

- Teste o novo trabalho/tempo de execução sem estado em um ambiente que não seja de produção antes de tentar uma atualização de produção.
- Considere primeiro testar a atualização com estado com um aplicativo que não seja de produção.
- Certifique-se de que seu novo gráfico de tarefas tenha um estado compatível com o instantâneo que você usará para iniciar seu aplicativo atualizado.
  - Certifique-se de que os tipos armazenados nos estados do operador permaneçam os mesmos. Se o tipo mudou, o Apache Flink não pode restaurar o estado do operador.
  - Certifique-se de que o Operador que IDs você definiu usando o `uid` método permaneça o mesmo. O Apache Flink tem uma forte recomendação para atribuir itens exclusivos IDs aos operadores. Para obter mais informações, consulte Atribuição de operador IDs na documentação do Apache Flink.

    Se você não atribuir IDs aos seus operadores, o Flink os gera automaticamente. Nesse caso, eles podem depender da estrutura do programa e, se alterados, podem causar problemas de compatibilidade. O Flink usa o Operador IDs para combinar o estado do instantâneo com o operador. A alteração do operador IDs faz com que o aplicativo não seja iniciado ou que o estado armazenado no instantâneo seja descartado e o novo operador seja iniciado sem estado.
  - Não altere a chave usada para armazenar o estado da chave.
  - Não modifique o tipo de entrada de operadores com estado, como janela ou junção. Isso altera implicitamente o tipo do estado interno do operador, causando uma incompatibilidade de estado.

Precauções e problemas conhecidos com atualizações de aplicativos

Limitações conhecidas da compatibilidade de estados

- Se você estiver usando a TabelaAPI, o Apache Flink não garante a compatibilidade de estado entre as versões do Flink. Para obter mais informações, consulte Stateful Upgrades and Evolution na documentação do Apache Flink.
- Os estados do Flink 1.6 não são compatíveis com o Flink 1.18. O API rejeita sua solicitação se você tentar atualizar da versão 1.6 para a versão 1.18 e posterior com o estado. Você pode atualizar para 1.8, 1.11, 1.13 e 1.15 e tirar um instantâneo e, em seguida, atualizar para 1.18 e

versões posteriores. Para obter mais informações, consulte Atualizando aplicativos e versões do Flink na documentação do Apache Flink.

Problemas conhecidos com o Flink Kinesis Connector

- Se você estiver usando o Flink 1.11 ou anterior e usando o `amazon-kinesis-connector-flink` conector para suporte a E nhanced-fan-out (EFO), deverá tomar medidas adicionais para uma atualização contínua para o Flink 1.13 ou posterior. Isso ocorre devido à alteração no nome do pacote do conector. Para obter mais informações, consulte amazon-kinesis-connector-flink.

  O `amazon-kinesis-connector-flink` conector para Flink 1.11 e versões anteriores usa a embalagem`software.amazon.kinesis`, enquanto o conector Kinesis para Flink 1.13 e versões posteriores usa. `org.apache.flink.streaming.connectors.kinesis` Use essa ferramenta para apoiar sua migração: amazon-kinesis-connector-flink-state-migrator.
- Se você estiver usando o Flink 1.13 ou anterior `FlinkKinesisProducer` e atualizando para o Flink 1.15 ou posterior, para uma atualização com estado, você deve continuar usando o Flink 1.15 ou posterior, `FlinkKinesisProducer` em vez do mais novo. `KinesisStreamsSink` No entanto, se você já tiver um `uid` conjunto personalizado no coletor, poderá alternar para `FlinkKinesisProducer` ele `KinesisStreamsSink` porque não mantém o estado. O Flink o tratará como o mesmo operador porque um personalizado foi `uid` definido.

Aplicativos Flink escritos em Scala

- A partir do Flink 1.15, o Apache Flink não inclui o Scala no tempo de execução. Você deve incluir a versão do Scala que deseja usar e outras dependências do Scala em seu código JAR /zip ao atualizar para o Flink 1.15 ou posterior. Para obter mais informações, consulte Amazon Managed Service for Apache Flink for Apache Flink versão 1.15.2.
- Se seu aplicativo usa o Scala e você o está atualizando do Flink 1.11 ou anterior (Scala 2.11) para o Flink 1.13 (Scala 2.12), certifique-se de que seu código use o Scala 2.12. Caso contrário, seu aplicativo Flink 1.13 pode falhar em encontrar classes Scala 2.11 no tempo de execução do Flink 1.13.

Coisas a considerar ao fazer o downgrade do aplicativo Flink

- O downgrade dos aplicativos Flink é possível, mas limitado aos casos em que o aplicativo estava sendo executado anteriormente com a versão mais antiga do Flink. Para uma atualização com

estado, o Managed Service para Apache Flink exigirá o uso de um snapshot tirado com uma versão correspondente ou anterior para o downgrade.

- Se você estiver atualizando seu tempo de execução do Flink 1.13 ou posterior para o Flink 1.11 ou anterior e se seu aplicativo usar o back-end de HashMap estado, seu aplicativo falhará continuamente.

Implemente o escalonamento de aplicativos no Managed Service para 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 informações sobre como o Apache Flink agenda instâncias paralelas de tarefas, consulte Execução paralela na documentação do Apache Flink.

Tópicos

Configure o paralelismo de aplicativos e ParallelismPer KPU

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 essa propriedade para definir o número de tarefas paralelas que podem ser agendadas por 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 de `ParallelismPerKPU` leva à utilização total dos KPU recursos.

> **Note**
>
> O limite para `Parallelism` é igual às `ParallelismPerKPU` vezes o limite para KPUs (que tem um padrão de 64). O KPUs limite pode ser aumentado solicitando 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 Configurando o paralelismo: Operador na documentação do Apache Flink.

Aloque unidades de processamento do Kinesis

O serviço gerenciado para Apache Flink provisiona a capacidade como. KPUs Um único KPU fornece 1 v CPU e 4 GB de memória. Para cada KPU alocação, também são fornecidos 50 GB de armazenamento de aplicativos em execução.

O Managed Service for Apache Flink calcula o KPUs que é necessário para executar seu aplicativo usando as `ParallelismPerKPU` propriedades `Parallelism` e, 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 Atualize o paralelismo do seu aplicativo.

O limite padrão 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**
>
> Um adicional KPU é cobrado para fins de orquestração. Para obter mais informações, consulte Preço do Managed Service for Apache Flink.

Atualize o paralelismo do seu aplicativo

Esta seção contém exemplos de solicitações de API ações que definem o paralelismo de um aplicativo. Para obter mais exemplos e instruções sobre como usar blocos de solicitação com API ações, consulteCódigo de exemplo do Managed Service for Apache Flink API.

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

```
{
   "ApplicationName": "string",
   "RuntimeEnvironment":"FLINK-1_18",
   "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"
         }
      }
   }
}
```

Use o escalonamento automático no Managed Service para Apache Flink

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 ajuste de escala automático está habilitado por padrão. O Managed Service for Apache Flink monitora o uso de recursos (CPU) do seu aplicativo e dimensiona elasticamente o paralelismo do seu aplicativo para cima ou para baixo de acordo:

- Seu aplicativo aumenta (aumenta o paralelismo) se o máximo da CloudWatch métrica `containerCPUUtilization` for maior que 75 por cento ou mais por 15 minutos. Isso significa que a `ScaleUp` ação é iniciada quando há 15 pontos de dados consecutivos com um período de 1 minuto igual ou superior a 75%. Uma `ScaleUp` ação dobra o valor `CurrentParallelism` do seu aplicativo. `ParallelismPerKPU`não foi modificado. Como consequência, o número de alocados KPUs também dobra.

- Seu aplicativo é reduzido (diminui o paralelismo) quando seu CPU uso permanece abaixo de 10% por seis horas. Isso significa que a `ScaleDown` ação é iniciada quando há 360 pontos de dados consecutivos com um período de 1 minuto inferior a 10%. Uma `ScaleDown` ação divide pela metade (arredondando para cima) o paralelismo do aplicativo. `ParallelismPerKPU`não é modificado, e o número de alocados KPUs também é reduzido pela metade (arredondado para cima).

> **Note**
>
> Um período máximo de `containerCPUUtilization` mais 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 inicializada.

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 API ação 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 AWS conta é cobrada pelas KPUs provisões do Managed Service for Apache Flink, que é uma função do seu aplicativo `parallelism` e `parallelismPerKPU` das configurações. 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 64KPUs. Para obter mais informações, consulte Serviço gerenciado para cota de notebooks Apache Flink e Studio.
- 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 seu aplicativo, consulte. the section called “Atualize o paralelismo do seu aplicativo”
  - 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 para Apache Flink”.

Implemente o escalonamento automático personalizado

Se você quiser um controle mais refinado sobre o escalonamento automático ou usar outras métricas de gatilho`containerCPUUtilization`, você pode usar este exemplo:

- AutoScaling

  Esses exemplos ilustram como escalar seu aplicativo Managed Service for Apache Flink usando uma CloudWatch métrica diferente da aplicação Apache Flink, incluindo métricas do Amazon e do Amazon Kinesis MSK Data Streams, usadas como fontes ou coletor.

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

Implemente o escalonamento automático programado

Se sua carga de trabalho seguir um perfil previsível ao longo do tempo, talvez você prefira escalar seu aplicativo Apache Flink preventivamente. Isso dimensiona seu aplicativo em um horário programado, em vez de escalar reativamente com base em uma métrica. Para configurar a escalabilidade para cima e para baixo em horários fixos do dia, você pode usar este exemplo:

- ScheduledScaling

maxParallelism considerações

O paralelismo máximo que uma tarefa do Flink pode escalar é limitado pelo mínimo `maxParallelism` em todos os operadores da tarefa. Por exemplo, se você tem um trabalho simples com apenas uma fonte e um coletor, e a fonte tem 16 e o coletor 8, o aplicativo não pode escalar além do paralelismo de 8. `maxParallelism`

Para saber como o padrão `maxParallelism` de um operador é calculado e como substituir o padrão, consulte Definindo o paralelismo máximo na documentação do Apache Flink.

Como regra básica, lembre-se de que, se você não definir `maxParallelism` para nenhum operador e iniciar seu aplicativo com um `maxParallelism` paralelismo menor ou igual a 128, todos os operadores terão 128.

> **Note**
>
> O paralelismo máximo do trabalho é o limite superior do paralelismo para escalar seu aplicativo mantendo o estado.
> Se você modificar `maxParallelism` um aplicativo existente, o aplicativo não poderá ser reiniciado a partir de um instantâneo anterior tirado com o antigo`maxParallelism`. Você só pode reiniciar o aplicativo sem um snapshot.
> Se você planeja escalar seu aplicativo para um paralelismo maior que 128, você deve definir explicitamente o `maxParallelism` em seu aplicativo.

- A lógica de escalonamento automático evitará escalar uma tarefa do Flink para um paralelismo que excederá o paralelismo máximo da tarefa.
- Se você usar um escalonamento automático personalizado ou escalabilidade programada, configure-os para que não excedam o paralelismo máximo do trabalho.
- Se você dimensionar manualmente seu aplicativo além do paralelismo máximo, o aplicativo falhará ao iniciar.

Adicionar tags ao Managed Service para aplicativos Apache Flink

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 adiciona tags ao criar um aplicativo usando o `tags` parâmetro da CreateApplicationação.

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 adiciona tags a um aplicativo usando a TagResourceação. Você não pode adicionar tags a um aplicativo usando a UpdateApplicationação.

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 as tags existentes, você usa a ListTagsForResourceação.

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, você usa a UntagResourceação.

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" ]
}
```

Use CloudFormation com o serviço gerenciado para Apache Flink

O exercício a seguir mostra como iniciar um aplicativo Flink criado com o AWS CloudFormation uso de uma função Lambda na mesma pilha.

Antes de começar

Antes de começar este exercício, siga as etapas para criar um aplicativo Flink usando AWS CloudFormation at AWS::KinesisAnalytics: :Application.

Escrever uma função Lambda

Para iniciar um aplicativo Flink após a criação ou atualização, usamos o aplicativo kinesisanalyticsv2 start-application. API A chamada será acionada por um AWS CloudFormation evento 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 processa eventos AWS CloudFormation recebidos, filtra tudo `Create` além disso `Update` e obtém o estado do aplicativo e o inicia se o estado for. `READY` Para obter o estado do aplicativo, você deve criar a função Lambda, conforme mostrado a seguir.

Crie uma função Lambda

Você cria uma função para que o Lambda “converse” com sucesso com o aplicativo e grave logs. Essa função usa políticas gerenciadas padrão, mas talvez você queira restringi-la ao uso de 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.

Invocar a função do Lambda

Agora, tudo o que resta é invocar a função do Lambda. Você faz isso 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.

Analise um exemplo estendido

O exemplo a seguir é uma versão ligeiramente estendida das etapas anteriores com um `RunConfiguration` ajuste 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_18'
      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 AWS de serviço usando AWS CloudFormation (Parte 1 de 3).
- Passo a passo: Pesquisando a Amazon Machine Image. IDs

Use o painel do Apache Flink com serviço gerenciado para o 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.

Acesse o painel do Apache Flink do seu aplicativo

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

Acesse o painel do Apache Flink do seu aplicativo usando o console 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 Managed Service for Apache Flink console, o URL que o console gera será válido por 12 horas.

Acesse o painel do Apache Flink do seu aplicativo usando o serviço gerenciado para o Apache Flink CLI

Você pode usar o Managed Service for Apache Flink CLI para gerar um URL para acessar o painel do seu aplicativo. O URL que você gera é válido por um período de tempo especificado.

> **Note**
>
> Se você não acessar o gerado URL em três minutos, ele não será mais válido.

Você gera seu painel URL usando a CreateApplicationPresignedUrlação. Você especifica os seguintes parâmetros para a ação:

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

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.

> **Note**
>
> Se você estiver usando uma versão do Apache Flink que está descontinuando, recomendamos que você atualize seu aplicativo para a versão mais recente compatível do Flink usando o Use atualizações de versão in-loco para o Apache Flink recurso do Managed Service for Apache Flink.

Versão Apache Flink	Status - Amazon Managed Service para Apache Flink	Status - Comunidade Apache Flink	Link
1.19.1	Compatível	Compatível	Amazon Managed Service para Apache Flink 1.19
1.18.1	Compatível	Compatível	Amazon Managed Service para Apache Flink 1.18
1.15.2	Compatível	Sem suporte	Amazon Managed Service para Apache Flink 1.15
1.13.1	Compatível	Sem suporte	Começando: Flink 1.13.2
1.11.1	Depreciando	Sem suporte	Informações sobre a versão anterior do Managed Service for Apache Flink(desconti

Versão Apache Flink	Status - Amazon Managed Service para Apache Flink	Status - Comunidade Apache Flink	Link
			nuando em 5 de novembro de 2024)
1.8.2	Depreciando	Sem suporte	Informações sobre a versão anterior do Managed Service for Apache Flink(descontinuando em 5 de novembro de 2024)
1.6.2	Depreciando	Sem suporte	Informações sobre a versão anterior do Managed Service for Apache Flink(descontinuando em 5 de novembro de 2024)

Tópicos

- Amazon Managed Service para Apache Flink 1.19
- Amazon Managed Service para Apache Flink 1.18
- Amazon Managed Service para Apache Flink 1.15
- Informações sobre a versão anterior do Managed Service for Apache Flink

Amazon Managed Service para Apache Flink 1.19

O Managed Service para Apache Flink agora oferece suporte ao Apache Flink versão 1.19.1. Esta seção apresenta os principais novos recursos e mudanças introduzidos com o Managed Service for Apache Flink, suporte ao Apache Flink 1.19.1.

> **Note**
>
> Se você estiver usando uma versão anterior compatível do Apache Flink e quiser atualizar seus aplicativos existentes para o Apache Flink 1.19.1, você pode fazer isso usando atualizações de versão do Apache Flink in-loco. Para obter mais informações, consulte Use atualizações de versão in-loco para o Apache Flink. Com as atualizações de versão no local, você mantém a rastreabilidade do aplicativo ARN em relação a uma única versão do Apache Flink, incluindo instantâneos, registros, métricas, tags, configurações do Flink e muito mais.

Atributos compatíveis

O Apache Flink 1.19.1 introduz melhorias no SQLAPI, como parâmetros nomeados, paralelismo de origem personalizado e estados diferentes para vários operadores do Flink. TTLs

Recursos suportados e documentação relacionada

Atributos compatíveis	Descrição	Referência da documentação do Apache Flink
SQLAPI: Support a configuração de estados diferentes TTLs usando o SQL Hint	Agora, os usuários podem configurar o estado TTL nas junções regulares do stream e na agregação de grupos.	FLIP-373: Configurando um estado diferente usando o Hint TTLs SQL
SQLAPI: Support parâmetros nomeados para funções e procedimentos de chamada	Agora, os usuários podem usar parâmetros nomeados em funções, em vez de confiar na ordem dos parâmetros.	FLIP-378: Support parâmetros nomeados para funções e procedimentos de chamada
SQLAPI: Definindo paralelismo para fontes SQL	Agora, os usuários podem especificar o paralelismo para as fontes. SQL	FLIP-367: Support Setting Parallelism for Table/Sources SQL
SQLAPI: Janela de sessão de suporte TVF	Agora, os usuários podem usar as funções com valor de tabela da janela de sessão.	FLINK-24024: Janela da sessão de suporte TVF

Atributos compatíveis	Descrição	Referência da documentação do Apache Flink
SQLAPI: A TVF agregação de janelas suporta entradas de registro de alterações	Agora, os usuários podem realizar a agregação de janelas nas entradas do changelog.	FLINK-20281: A agregação de janelas suporta a entrada de fluxo do changelog
Support Python 3.11	O Flink agora suporta o Python 3.11, que é 10 a 60% mais rápido em comparação com o Python 3.10. Para obter mais informações, consulte O que há de novo no Python 3.11.	FLINK-33030: Adicionar suporte ao python 3.11
Forneça métricas para o TwoPhaseCommitting coletor	Os usuários podem visualizar estatísticas sobre o status dos committers em coletores de confirmação em duas fases.	FLIP-371: Forneça contexto de inicialização para a criação do Committer em TwoPhaseCommittingSink
Rastreie Reporters para reinício do trabalho e verificação	Agora, os usuários podem monitorar os rastreamentos da duração do ponto de verificação e das tendências de recuperação. No Amazon Managed Service para Apache Flink, habilitamos os relatores de rastreamento SLF4j por padrão, para que os usuários possam monitorar pontos de verificação e rastreamentos de trabalhos por meio de registros de aplicativos. CloudWatch	FLIP-384: Introduza TraceReporter e use-o para criar rastreamentos de verificação e recuperação

> **Note**
>
> Você pode optar pelos seguintes recursos enviando um caso de suporte:

Recursos de aceitação e documentação relacionada

Funcionalidades de adesão	Descrição	Referência da documentação do Apache Flink
Support usando um intervalo maior de ponto de verificação quando a fonte está processando a lista de pendências	Esse é um recurso opcional, pois os usuários devem ajustar a configuração de acordo com seus requisitos específicos de trabalho.	FLIP-309: Support usando um intervalo maior de checkpoint quando a fonte está processando a lista de pendências
Redirecione System.out e System.err para registros Java	Esse é um recurso opcional. No Amazon Managed Service para Apache Flink, o comportamento padrão é ignorar a saída de System.out e System.err porque a melhor prática na produção é usar o registrador Java nativo.	FLIP-390: Support System desligado e erro ao ser redirecionado ou descartado LOG

Para a documentação da versão 1.19.1 do Apache Flink, consulte a documentação do Apache Flink v1.19.1.

Alterações no Amazon Managed Service para Apache Flink 1.19.1

O Logging Trace Reporter ativado por padrão

O Apache Flink 1.19.1 introduziu pontos de verificação e rastreamentos de recuperação, permitindo que os usuários depurassem melhor os problemas de pontos de verificação e recuperação de tarefas. No Amazon Managed Service para Apache Flink, esses rastreamentos são registrados no fluxo de CloudWatch log, permitindo que os usuários detalhem o tempo gasto na inicialização do trabalho e registrem o tamanho histórico dos pontos de verificação.

A estratégia de reinicialização padrão agora é atraso exponencial

No Apache Flink 1.19.1, há melhorias significativas na estratégia de reinicialização com atraso exponencial. No Amazon Managed Service para Apache Flink a partir do Flink 1.19.1, as tarefas do Flink usam a estratégia de reinicialização com atraso exponencial por padrão. Isso significa que as tarefas do usuário se recuperarão mais rapidamente de erros transitórios, mas não sobrecarregarão os sistemas externos se as reinicializações das tarefas persistirem.

Componentes

Componente	Version (Versão)
Java	11 (recomendado)
Python	3.11
Kinesis Data Analytics Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
Connectors	Para obter informações sobre os conectores disponíveis, consulte Conectores Apache Flink.
Apache Beam (somente aplicativos Beam)	Não há Apache Flink Runner compatível para o Flink 1.19. Para obter mais informações, consulte Compatibilidade de versão do Flink.

Problemas conhecidos

Feixe Apache

Atualmente, não há Apache Flink Runner compatível para Flink 1.19 no Apache Beam. Para obter mais informações, consulte Compatibilidade de versão do Flink.

Amazon Managed Service para Apache Flink Studio

O Studio usa notebooks 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. É necessário atualizar o Flink Interpreter do Zeppelin

para permitir o suporte ao Flink 1.19. Este trabalho está agendado com a comunidade do Zeppelin e atualizaremos essas notas quando estiver concluído. Você pode continuar usando o Flink 1.15 com o Amazon Managed Service para Apache Flink Studio. Para obter mais informações, consulte Criação de um notebook Studio.

Amazon Managed Service para Apache Flink 1.18

O Managed Service para Apache Flink agora oferece suporte ao Apache Flink versão 1.18.1. Conheça os principais novos recursos e mudanças introduzidos com o Managed Service for Apache Flink, suporte ao Apache Flink 1.18.1.

> **Note**
>
> Se você estiver usando uma versão anterior compatível do Apache Flink e quiser atualizar seus aplicativos existentes para o Apache Flink 1.18.1, você pode fazer isso usando atualizações de versão do Apache Flink in-loco. Com as atualizações de versão no local, você mantém a rastreabilidade do aplicativo ARN em relação a uma única versão do Apache Flink, incluindo instantâneos, registros, métricas, tags, configurações do Flink e muito mais. Você pode usar esse recurso em `RUNNING` um `READY` estado. Para obter mais informações, consulte Use atualizações de versão in-loco para o Apache Flink.

Atributos compatíveis	Descrição	Referência da documentação do Apache Flink
Conector Opensearch	Esse conector inclui um coletor que fornece at-least-once garantias.	github: Conector Opensearch
Conector Amazon DynamoDB	Esse conector inclui um coletor que fornece at-least-once garantias.	Coletor do Amazon DynamoDB
Conector MongoDB	Esse conector inclui uma fonte e um coletor que fornecem at-least-once garantias.	Conector MongoDB

Atributos compatíveis	Descrição	Referência da documentação do Apache Flink
Separe o Hive com o planejador Flink	Você pode usar o dialeto Hive diretamente sem a troca extraJAR.	FLINK-26603: Separe o Hive com o planejador Flink
Desativar WAL em R ocksDBWrite BatchWrapper por padrão	Isso proporciona tempos de recuperação mais rápidos.	FLINK-32326: Desativar em WAL R por padrão ocksDBWri te BatchWrapper
Melhore o desempenho da agregação de marcas d'água ao ativar o alinhamento da marca d'água	Melhora o desempenho da agregação de marcas d'água ao ativar o alinhamento da marca d'água e adiciona o benchmark relacionado.	FLINK-32524: Desempenh o de agregação de marcas d'água
Prepare o alinhamento da marca d'água para uso em produção	Elimina o risco de sobrecarg a de grandes trabalhos JobManager	FLINK-32548: Preparar o alinhamento da marca d'água
Configurável RateLimit ingStratey para coletor assíncrono	RateLimitingStrategy permite que você configure a decisão sobre o que escalar, quando escalar e quanto escalar.	FLIP-242: Introdução configurável RateLimit ingStrategy para Async Sink
Estatísticas de tabelas e colunas de busca em massa	Desempenho aprimorado da consulta.	FLIP-247: Busca em massa de estatísticas de tabelas e colunas para determinadas partições

Para a documentação de lançamento do Apache Flink 1.18.1, consulte Anúncio de lançamento do Apache Flink 1.18.1.

Mudanças no Amazon Managed Service para Apache Flink com Apache Flink 1.18

Akka substituída por Pekko

O Apache Flink substituiu Akka por Pekko no Apache Flink 1.18. Essa alteração é totalmente suportada no Managed Service for Apache Flink a partir do Apache Flink 1.18.1 e versões posteriores. Você não precisa modificar seus aplicativos como resultado dessa alteração. Para obter mais informações, consulte FLINK-32468: Substitua Akka por Pekko.

Support a PyFlink execução do Runtime no Modo Thread

Essa alteração do Apache Flink introduz um novo modo de execução para a estrutura Pyflink Runtime, o Process Mode. O Modo de Processo agora pode executar funções definidas pelo usuário do Python no mesmo encadeamento em vez de em um processo separado.

Componentes

Componente	Version (Versão)
Java	11 (recomendado)
Scala	Desde a versão 1.15, o Flink é independente de Scala. Para referência, o MSF Flink 1.18 foi verificado em relação ao Scala 3.3 (). LTS
Serviço gerenciado para Apache Flink Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
AWS Kinesis Connector (flink-connector-kinesis) [Fonte]	4.2.0-1.18
AWS Conector Kinesis (flink-connector-kinesis) [Pia]	4.2.0-1.18
Apache Beam (somente aplicativos Beam)	Anterior e até a versão 2.75.0. Para obter mais informações, consulte Compatibilidade de versão do Flink.

Correções de erros

Compressão de estado no Apache Flink 1.18.1

O Apache Flink oferece compressão opcional (padrão: desligado) para todos os pontos de verificação e pontos de salvamento. O Apache Flink identificou um bug no Flink 1.18.1 em que o estado do operador não pôde ser restaurado adequadamente quando a compactação de instantâneos foi ativada. Isso pode resultar na perda de dados ou na incapacidade de restaurar a partir do ponto de verificação. Para obter mais informações, consulte FLINK-34063: Quando a compactação de instantâneos está ativada, o redimensionamento de um operador de origem faz com que algumas divisões sejam perdidas.

Para resolver isso, o Amazon Managed Service para Apache Flink fez o backport da correção que será incluída nas futuras versões do Apache Flink. Para obter mais informações, consulte github: Sempre liberte os buffers de compressão.

Problemas conhecidos

Amazon Managed Service para Apache Flink Studio

O Studio usa notebooks 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. É necessário atualizar o Flink Interpreter do Zeppelin para permitir o suporte ao Flink 1.18. Este trabalho está agendado com a comunidade do Zeppelin e atualizaremos essas notas quando estiver concluído. Você pode continuar usando o Flink 1.15 com o Amazon Managed Service para Apache Flink Studio. Para obter mais informações, consulte Criação de um notebook Studio.

Amazon Managed Service para Apache Flink 1.15

O Managed Service para Apache Flink oferece suporte aos seguintes novos recursos no Apache 1.15.2:

Atributo	Descrição	Referência do Apache FLIP
Coletor assíncrono	Uma estrutura AWS contribuí da para criar destinos assíncronos que permite aos desenvolvedores criar AWS conectores personalizados com menos da metade do esforço anterior. Para obter	FLIP-171: Coletor assíncrono.

Atributo	Descrição	Referência do Apache FLIP
	mais informações, consulte The Generic Asynchronous Base Sink.	
Coletor do Kinesis Data Firehose	AWS contribuiu com um novo Amazon Kinesis Firehose Sink usando a estrutura Async.	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 Job com o Savepoint.
Desacoplamento do Scala	Agora, os usuários podem aproveitar o Java API 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 de Scala.
Scala	Veja desacoplamento do Scala acima	FLIP-28: Objetivo de longo prazo de tornar o Flink-table livre de Scala.

Atributo	Descrição	Referência do Apache FLIP
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 métricas de conectores 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: Support Checkpoints 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. Use um notebook Studio com serviço gerenciado para Apache Flink

EFOconector

Ao atualizar para o Managed Service for Apache Flink versão 1.15, verifique se você está usando o EFO conector 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 fazer o upgrade para o Amazon Managed Service for Apache Flink for Apache Flink versão 1.15, verifique se você está usando o conector Kafka mais recente. APIs O Apache Flink foi descontinuado FlinkKafkaConsumere o These APIs for the Kafka sink não pode se FlinkKafkaProducercomprometer com o Kafka para o Flink 1.15. Certifique-se de que você está usando KafkaSourceKafkaSinke.

Componentes

Componente	Version (Versão)
Java	11 (recomendado)
Scala	2.12
Serviço gerenciado para Apache Flink 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

Informações sobre a versão anterior do Managed Service for Apache Flink

> **Note**
>
> As versões 1.6, 1.8 e 1.11 do Apache Flink não são suportadas pela comunidade Apache Flink há mais de três anos. Planejamos descontinuar essas versões no Amazon Managed Service para Apache Flink em 5 de novembro de 2024. A partir dessa data, você não poderá criar novos aplicativos para essas versões do Flink. Você pode continuar executando os aplicativos existentes no momento. Você pode atualizar seus aplicativos de forma estável usando o recurso de atualizações de versão in-loco no Amazon Managed Service para Apache Flink. Para obter mais informações, consulte. Use atualizações de versão in-loco para o Apache Flink

As versões 1.15.2 e 1.13.2 do Apache Flink são suportadas pelo Managed Service for Apache Flink, mas não são mais suportadas pela comunidade Apache Flink.

Este tópico contém as seguintes seções:

Usando o conector Apache Flink Kinesis Streams 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`.

Criando aplicativos com o Apache Flink 1.8.2

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 1.8.2.

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

Componente	Version (Versão)
Java	1.8 (recomendado)
Apache Flink	1.8.2
Serviço gerenciado para Apache Flink for Flink Runtime () aws-kinesisanalytics-runtime	1.0.1
Serviço gerenciado para conectores Apache Flink 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. Crie um serviço gerenciado para o aplicativo Apache Flink

Criando aplicativos 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	Version (Versão)
Java	1.8 (recomendado)
AWS SDK para Java	1.11.379
Apache Flink	1.6.2
Serviço gerenciado para Apache Flink for Flink Runtime () aws-kinesisanalytics-runtime	1.0.1
Serviço gerenciado para conectores Apache Flink 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. Crie um serviço gerenciado para o aplicativo Apache Flink

Atualizando aplicativos

Para atualizar a versão Apache Flink de um aplicativo Amazon Managed Service para Apache Flink, use o recurso de atualização de versão do Apache Flink no local usando o AWS CLI SDK ou o. AWS AWS CloudFormation AWS Management Console Para ter mais informações, consulte Use atualizações de versão in-loco para o Apache Flink.

Você pode usar esse recurso com qualquer aplicativo existente que você usa com o Amazon Managed Service para Apache Flink em `READY` ou `RUNNING` estado.

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).

Começando: 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 serviço gerenciado para o aplicativo Apache Flink
- Pré-requisitos para concluir os exercícios
- Etapa 1: configurar uma AWS conta e criar um usuário administrador
- Próxima etapa
- Etapa 2: configurar o AWS Command Line Interface (AWS CLI)

- Etapa 3: criar e executar um serviço gerenciado para o aplicativo Apache Flink
- Etapa 4: limpar AWS os recursos
- Etapa 5: Próximas etapas

Componentes de um serviço gerenciado para o aplicativo 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 Adicione fontes de dados de streaming ao Managed Service for Apache Flink.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Transforme dados usando operadores no Managed Service for Apache Flink.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector de coletor grava dados em um stream de dados do Kinesis, um stream Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

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 Configurar uma AWS conta e criar um usuário administrador.

Etapa 1: configurar uma AWS conta e criar um usuário administrador

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

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

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

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. 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 com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira 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. Habilite a autenticação multifator (MFA) para o usuário raiz.

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

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

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

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando 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 AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos 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 AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credencia is de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Próxima etapa

Configure o AWS Command Line Interface (AWS CLI)

Próxima etapa

Etapa 2: configurar o AWS Command Line Interface (AWS CLI)

Etapa 2: configurar o AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura o AWS CLI para usar com o Managed Service para 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 tem o AWS CLI instalado, talvez seja necessário fazer o upgrade para obter a funcionalidade mais recente. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface . Para verificar a versão do AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios deste tutorial exigem a seguinte AWS CLI versão ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar o 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 AWS CLI `config` arquivo. 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 AWS regiões 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 a AWS 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 serviço gerenciado para o aplicativo Apache Flink

Etapa 3: criar e executar um serviço gerenciado para o aplicativo 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:

- Crie dois streams de dados do Amazon Kinesis
- Grave registros de amostra no fluxo de entrada
- Baixe e examine o código Java de streaming do Apache Flink
- Compilar o código do aplicativo
- Faça o upload do código Java de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Próxima etapa

Crie dois streams de dados do Amazon 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 stream (`ExampleInputStream`), use o seguinte comando do Amazon Kinesis `create-stream` AWS CLI .

```
$ 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
```

Grave 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
```

Baixe e examine o código Java de streaming do Apache Flink

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 do runtime, consulte Use propriedades de tempo de execução no Managed Service para Apache Flink.

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 Cumpra os 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 o AWS CLI, especifique o tipo de conteúdo do 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 de streaming do Apache Flink

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 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. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Selecionar arquivos, selecione 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.

Crie e execute o aplicativo 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. Ao criar o aplicativo usando o AWS CLI, você cria esses recursos separadamente.

Tópicos

Crie e execute 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`

Edite 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. 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. Selecione 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`

Execute 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.

Pare o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualizar 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).

Crie e execute o aplicativo (AWS CLI)

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

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 `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 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 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 “Criação de 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.

Crie o serviço gerenciado para o aplicativo 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 “Configurar o registro de aplicativos no Managed Service para Apache Flink”.

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 de código, use a `UpdateApplication` AWS CLI ação.

> **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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 “Crie dois streams de dados do Amazon Kinesis”.

```
{
    "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 AWS os recursos

Etapa 4: limpar AWS os recursos

Esta seção inclui procedimentos para limpar AWS os recursos criados no tutorial de introdução.

Este tópico contém as seguintes seções:

Exclua 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.

Exclua seus streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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 AWS de streaming de dados para o Amazon Kinesis: A solução de dados de AWS streaming para o Amazon Kinesis configura automaticamente AWS os serviços necessários para

capturar, armazenar, processar e entregar dados de streaming 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.

- AWS Solução de streaming de dados para Amazon MSK: A solução AWS de dados de streaming para Amazon MSK fornece AWS CloudFormation modelos em que os dados fluem por produtores, armazenamento de streaming, 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.
- 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 - descontinuando

> **Note**
>
> As versões 1.6, 1.8 e 1.11 do Apache Flink não são suportadas pela comunidade Apache Flink há mais de três anos. Planejamos descontinuar essas versões no Amazon Managed Service para Apache Flink em 5 de novembro de 2024. A partir dessa data, você não poderá

criar novos aplicativos para essas versões do Flink. Você pode continuar executando os aplicativos existentes no momento. Você pode atualizar seus aplicativos de forma estável usando o recurso de atualizações de versão in-loco no Amazon Managed Service para Apache Flink. Para obter mais informações, consulte. Use atualizações de versão in-loco para o Apache Flink

Este tópico contém uma versão do Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink 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 serviço gerenciado para o aplicativo Apache Flink
- Pré-requisitos para concluir os exercícios
- Etapa 1: configurar uma AWS conta e criar um usuário administrador
- Etapa 2: Configurar a AWS Command Line Interface (AWS CLI)
- Etapa 3: criar e executar um serviço gerenciado para o aplicativo Apache Flink
- Etapa 4: limpar AWS os recursos
- Etapa 5: Próximas etapas

Componentes de um serviço gerenciado para o aplicativo 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 Adicione fontes de dados de streaming ao Managed Service for Apache Flink.

- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Transforme dados usando operadores no Managed Service for Apache Flink.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector de coletor grava dados em um stream de dados do Kinesis, um stream Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

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 Configurar uma AWS conta e criar um usuário administrador.

Etapa 1: configurar uma AWS conta e criar um usuário administrador

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

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

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

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. 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 com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira 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. Habilite a autenticação multifator (MFA) para o usuário raiz.

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

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .

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

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

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

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando 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 temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferrament as e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credencia is de longo prazo no Guia

Qual usuário precisa de acesso programático?	Para	Por
		de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Próxima etapa

Configure o AWS Command Line Interface (AWS CLI)

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

Nesta etapa, você baixa e configura o AWS CLI para usar com o Managed Service para 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 tem o AWS CLI instalado, talvez seja necessário fazer o upgrade para obter a funcionalidade mais recente. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface . Para verificar a versão do AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios deste tutorial exigem a seguinte AWS CLI versão ou posterior:

```
aws-cli/1.16.63
```

Para configurar o 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 AWS CLI `config` arquivo. 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 AWS regiões 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 a AWS 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 serviço gerenciado para o aplicativo Apache Flink

Etapa 3: criar e executar um serviço gerenciado para o aplicativo 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:

Crie dois streams de dados do Amazon 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 stream (`ExampleInputStream`), use o seguinte comando do Amazon Kinesis `create-stream` AWS CLI .

```
$ 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
```

Grave 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
```

Baixe e examine o código Java de streaming do Apache Flink

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 do runtime, consulte Use propriedades de tempo de execução no Managed Service para Apache Flink.

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 Cumpra os 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 o AWS CLI, especifique o tipo de conteúdo do 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 de streaming do Apache Flink

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 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. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Selecionar arquivos, selecione 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.

Crie e execute o aplicativo 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. Ao criar o aplicativo usando o AWS CLI, você cria esses recursos separadamente.

Tópicos

Crie e execute 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, 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`

Edite 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. 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. 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**
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. Selecione 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`

Execute 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.

Pare o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualizar 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).

Crie e execute 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.

Crie o serviço gerenciado para o aplicativo 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.

Inicie 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.

Pare 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 “Configurar o registro de aplicativos no Managed Service para Apache Flink”.

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 de código, use a `UpdateApplication` AWS CLI ação.

> **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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 “Crie dois streams de dados do Amazon Kinesis”.

```
{
    "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 AWS os recursos

Etapa 4: limpar AWS os recursos

Esta seção inclui procedimentos para limpar AWS os recursos criados no tutorial de introdução.

Este tópico contém as seguintes seções:

Exclua 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.

Exclua seus streams 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.

Exclua 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.

Exclua quatro recursos do IAM

1. Abra o console do 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 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 AWS de streaming de dados para o Amazon Kinesis: A solução de dados de AWS streaming para o Amazon Kinesis configura automaticamente AWS os serviços necessários para capturar, armazenar, processar e entregar dados de streaming 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.
- AWS Solução de streaming de dados para Amazon MSK: A solução AWS de dados de streaming para Amazon MSK fornece AWS CloudFormation modelos em que os dados fluem por produtores, armazenamento de streaming, 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.
- 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.

Começando: Flink 1.8.2 - descontinuando

> **Note**
>
> As versões 1.6, 1.8 e 1.11 do Apache Flink não são suportadas pela comunidade Apache Flink há mais de três anos. Planejamos descontinuar essas versões no Amazon Managed Service para Apache Flink em 5 de novembro de 2024. A partir dessa data, você não poderá criar novos aplicativos para essas versões do Flink. Você pode continuar executando os aplicativos existentes no momento. Você pode atualizar seus aplicativos de forma estável usando o recurso de atualizações de versão in-loco no Amazon Managed Service para Apache Flink. Para obter mais informações, consulte. Use atualizações de versão in-loco para o Apache Flink

Este tópico contém uma versão do Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink tutorial que usa o Apache Flink 1.8.2.

Tópicos

Componentes do serviço gerenciado para o aplicativo 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 Adicione fontes de dados de streaming ao Managed Service for Apache Flink.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Transforme dados usando operadores no Managed Service for Apache Flink.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector de coletor grava dados em um stream de dados do Kinesis, um stream Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

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 Usando o conector Apache Flink Kinesis Streams 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 AWS conta e criar um usuário administrador.

Etapa 1: configurar uma AWS conta e criar um usuário administrador

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

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

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

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. 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 com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira 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. Habilite a autenticação multifator (MFA) para o usuário raiz.

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

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

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

Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .
2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando 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 AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM

Qual usuário precisa de acesso programático?	Para	Por
		Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credencia is de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Etapa 2: configurar o AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura o AWS CLI para usar com o Managed Service para 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 tem o AWS CLI instalado, talvez seja necessário fazer o upgrade para obter a funcionalidade mais recente. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface . Para verificar a versão do AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios deste tutorial exigem a seguinte AWS CLI versão ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar o 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 AWS CLI `config` arquivo. 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 AWS 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 a AWS 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 serviço gerenciado para o aplicativo Apache Flink

Etapa 3: criar e executar um serviço gerenciado para o aplicativo 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:

- Crie dois streams de dados do Amazon Kinesis
- Grave registros de amostra no fluxo de entrada
- Baixe e examine o código Java de streaming do Apache Flink
- Compilar o código do aplicativo
- Faça o upload do código Java de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Próxima etapa

Crie dois streams de dados do Amazon 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 stream (`ExampleInputStream`), use o seguinte comando do Amazon Kinesis `create-stream` AWS CLI .

```
$ 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
```

Grave 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
```

Baixe e examine o código Java de streaming do Apache Flink

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 do runtime, consulte Use propriedades de tempo de execução no Managed Service para Apache Flink.

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 “Usando o conector Apache Flink Kinesis Streams 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 o AWS CLI, especifique o tipo de conteúdo do 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 de streaming do Apache Flink

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 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. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Selecionar arquivos, selecione 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.

Crie e execute o aplicativo 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. Ao criar o aplicativo usando o AWS CLI, você cria esses recursos separadamente.

Tópicos

Crie e execute 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`*

Edite 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. 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. Selecione 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`

Execute 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.

Pare o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualizar 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).

Crie e execute o aplicativo (AWS CLI)

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

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 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 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.

Crie o serviço gerenciado para o aplicativo 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.

Inicie 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.

Pare 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 “Configurar o registro de aplicativos no Managed Service para Apache Flink”.

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 de código, use a `UpdateApplication` AWS CLI ação.

> **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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 “Crie dois streams de dados do Amazon Kinesis”.

```
{
    "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 AWS os recursos

Etapa 4: limpar AWS os recursos

Esta seção inclui procedimentos para limpar AWS os recursos criados no tutorial de introdução.

Este tópico contém as seguintes seções:

Exclua 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.

Exclua seus streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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.

Começando: Flink 1.6.2 - descontinuando

> **Note**
>
> As versões 1.6, 1.8 e 1.11 do Apache Flink não são suportadas pela comunidade Apache Flink há mais de três anos. Planejamos descontinuar essas versões no Amazon Managed Service para Apache Flink em 5 de novembro de 2024. A partir dessa data, você não poderá criar novos aplicativos para essas versões do Flink. Você pode continuar executando os aplicativos existentes no momento. Você pode atualizar seus aplicativos de forma estável usando o recurso de atualizações de versão in-loco no Amazon Managed Service para Apache Flink. Para obter mais informações, consulte. Use atualizações de versão in-loco para o Apache Flink

Este tópico contém uma versão do Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink tutorial que usa o Apache Flink 1.6.2.

Tópicos

- Componentes de um serviço gerenciado para o aplicativo Apache Flink
- Pré-requisitos para concluir os exercícios
- Etapa 1: configurar uma AWS conta e criar um usuário administrador

- Etapa 2: configurar o AWS Command Line Interface (AWS CLI)
- Etapa 3: criar e executar um serviço gerenciado para o aplicativo Apache Flink
- Etapa 4: limpar AWS os recursos

Componentes de um serviço gerenciado para o aplicativo 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 Adicione fontes de dados de streaming ao Managed Service for Apache Flink.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para ter mais informações, consulte Transforme dados usando operadores no Managed Service for Apache Flink.
- Coletor: o aplicativo produz dados para fontes externas usando coletores. Um conector de coletor grava dados em um stream de dados do Kinesis, um stream Firehose, um bucket do Amazon S3 etc. Para ter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

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 AWS conta e criar um usuário administrador.

Etapa 1: configurar uma AWS conta e criar um usuário administrador

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

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

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

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. 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 com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira 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. Habilite a autenticação multifator (MFA) para o usuário raiz.

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

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

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

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando 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 temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferrament as e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos 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 AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credencia is de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Etapa 2: configurar o AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura o AWS CLI para usar com um serviço gerenciado para o 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 tem o AWS CLI instalado, talvez seja necessário fazer o upgrade para obter a funcionalidade mais recente. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface . Para verificar a versão do AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios deste tutorial exigem a seguinte AWS CLI versão ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar o 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 AWS CLI `config` arquivo. 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 AWS regiões 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 a AWS 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 serviço gerenciado para o aplicativo Apache Flink

Etapa 3: criar e executar um serviço gerenciado para o aplicativo 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:

- Crie dois streams de dados do Amazon Kinesis
- Grave registros de amostra no fluxo de entrada
- Baixe e examine o código Java de streaming do Apache Flink
- Compilar o código do aplicativo
- Faça o upload do código Java de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink

Crie dois streams de dados do Amazon 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 stream (`ExampleInputStream`), use o seguinte comando do Amazon Kinesis `create-stream` AWS CLI .

```
$ 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
```

Grave 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
```

Baixe e examine o código Java de streaming do Apache Flink

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 do runtime, consulte Use propriedades de tempo de execução no Managed Service para Apache Flink.

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 “Especifique 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 o AWS CLI, especifique o tipo de conteúdo do 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 de streaming do Apache Flink

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 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. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
8. Na etapa Selecionar arquivos, selecione 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.

Crie e execute o aplicativo 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. Ao criar o aplicativo usando o AWS CLI, você cria esses recursos separadamente.

Tópicos

- Crie e execute o aplicativo (console)
- Crie e execute o aplicativo (AWS CLI)

Crie e execute 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`*

Edite 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. 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. Selecione 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`

Execute 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.

Pare o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualizar 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).

Crie e execute o aplicativo (AWS CLI)

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

Criação de 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 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 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 “Criação de 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.

Crie o serviço gerenciado para o aplicativo 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.

Inicie 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.

Pare 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 “Configurar o registro de aplicativos no Managed Service para Apache Flink”.

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 de código, use a `UpdateApplication` AWS CLI ação.

Para usar o AWS CLI, exclua seu pacote de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo bucket e nome de objeto 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 “Crie dois streams de dados do Amazon Kinesis”.

```
{
    "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 AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial de introdução.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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.

Exclua seus streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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.

Exemplos de versões anteriores (antigas) do Managed Service for Apache Flink

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

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 funciona
- Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink

> **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
- Exemplos de Scala

DataStream Exemplos de API

Os exemplos a seguir demonstram como criar aplicativos usando a API Apache Flink DataStream .

Tópicos

- Exemplo: janela caindo
- Exemplo: janela deslizante
- Exemplo: gravação em um bucket do Amazon S3
- Tutorial: Usando um serviço gerenciado para o aplicativo Apache Flink para replicar dados de um tópico em um cluster MSK para outro em uma VPC
- Exemplo: use um consumidor EFO com um stream de dados do Kinesis
- Exemplo: Escrevendo para o Firehose
- Exemplo: Leia de um stream do Kinesis em uma conta diferente
- Tutorial: Usando um armazenamento confiável personalizado com o Amazon MSK

Exemplo: janela caindo

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave 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 de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Limpar recursos da AWS

Crie 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>`**.

Grave 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 Preencha os pré-requisitos necessários no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.
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 de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Selecionar arquivos, selecione 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.

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

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`

Edite 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. 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. 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. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Execute 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 recursos da AWS

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Tumbling Window.

Este tópico contém as seguintes seções:

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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 obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave 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 de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Limpe AWS os recursos

Crie 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>`**.

Grave 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 Preencha os pré-requisitos necessários no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.
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 de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket e, em seguida, escolha Upload. <username>
2. Na etapa Selecionar arquivos, selecione 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.

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

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`*

Edite 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. 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. Selecione 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.

Configurar 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.

Execute 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.

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Sliding Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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: gravação 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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave 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 de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Verifique a saída do aplicativo
- Opcional: personalize a fonte e o coletor
- Limpe AWS os recursos

Crie 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`.

Grave 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 Preencha os pré-requisitos necessários no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.
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 de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie 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 Selecionar arquivos, selecione 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.

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

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`

Edite a política do IAM

Edite a política do IAM para adicionar permissões de acesso ao fluxo de dados do Kinesis.

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. 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. 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. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Execute 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 de streaming do Apache Flink” 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:

- Configurar o particionamento de dados
- Configurar a frequência de leitura
- Configurar o buffer de gravação

Configurar 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.

Configurar 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");
```

Configurar 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.

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos que você criou no tutorial do Amazon S3.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seu stream de dados do Kinesis
- Exclua seus objetos e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 stream 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.

Exclua seus recursos do IAM

1. Abra o console do 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 serviço gerenciado para o aplicativo Apache Flink para replicar dados de um tópico em um cluster MSK para outro em uma VPC

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Este tutorial contém as seguintes seções:

- Crie uma Amazon VPC com um cluster Amazon MSK
- Crie o código do aplicativo
- Faça o upload do código Java de streaming do Apache Flink
- Criar o aplicativo
- Configurar o aplicativo
- Execute o aplicativo
- Teste o aplicativo

Crie uma Amazon VPC com um cluster 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 AWS KafkaTutorialTopicDestination
```

- 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 AWS região 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 de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

> **Note**
>
> Se você excluiu o bucket do Amazon S3 no tutorial de introdução, siga a etapa the section called “Faça o upload do JAR arquivo de código do aplicativo” novamente.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Selecionar arquivos, selecione 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	AWS KafkaTutorialTopic
`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	AWS KafkaTutorialTopic Destination
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 Use propriedades de tempo de execução no Managed Service para Apache Flink.

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.

Execute 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.

Teste 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 AWS KafkaTutorialTopicDestination --from-
beginning
```

Se nenhum registro aparecer no tópico de destino, consulte a seção Não é possível acessar recursos em um VPC no tópico Solucionar problemas de serviço gerenciado para Apache Flink.

Exemplo: use um consumidor EFO com um stream de dados do Kinesis

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

Neste exercício, você cria um serviço gerenciado para o aplicativo Apache Flink que lê um stream de dados do Kinesis usando um consumidor de 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 conector AWS Kinesis (flink-connector-kinesis) 1.15.3.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave 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 de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink

- Limpe AWS os recursos

Crie 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>`**.

Grave 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 Preencha os pré-requisitos necessários no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.
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 de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie recursos dependentes.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Selecionar arquivos, selecione 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.

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

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`

Edite 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. 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. 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. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Execute 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

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial efo Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Excluir seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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: Escrevendo para o Firehose

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

Neste exercício, você cria um serviço gerenciado para o aplicativo Apache Flink que tem um stream de dados do Kinesis como fonte e um stream do 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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Esta seção contém as seguintes etapas:

- Crie recursos dependentes
- Grave registros de amostra no fluxo de entrada
- Baixe e examine o código Java de streaming do Apache Flink
- Compilar o código do aplicativo
- Faça o upload do código Java de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Limpar recursos da AWS

Crie 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 stream do Firehose no qual o aplicativo grava a saída ()`ExampleDeliveryStream`.

- Um bucket do Amazon S3 para armazenar o código do aplicativo (`ka-app-code-<username>`)

Você pode criar o stream do Kinesis, os buckets do Amazon S3 e o stream do 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`**.
- Criação de um stream de entrega do Amazon Kinesis Data Firehose no Guia do desenvolvedor do Amazon Data Firehose. Dê um nome ao seu stream do Firehose. **`ExampleDeliveryStream`** Ao criar o stream do Firehose, crie também o destino S3 e a função do IAM do stream Firehose.
- 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>`**.

Grave 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 Java de streaming do Apache Flink

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 Firehose para gravar dados em um stream do Firehose. O trecho a seguir cria o coletor Firehose:

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {
        Properties sinkProperties = new Properties();
        sinkProperties.setProperty(AWS_REGION, region);

        return KinesisFirehoseSink.<String>builder()
                .setFirehoseClientProperties(sinkProperties)
                .setSerializationSchema(new SimpleStringSchema())
                .setDeliveryStreamName(outputDeliveryStreamName)
                .build();
    }
```

Compilar 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 Preencha os pré-requisitos necessários no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.
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 “Usando o conector Apache Flink Kinesis Streams 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 de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie 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 Selecionar arquivos, selecione 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.

Crie e execute o aplicativo 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. Ao criar o aplicativo usando o AWS CLI, você cria esses recursos separadamente.

Tópicos

- Crie e execute o aplicativo (console)
- Crie e execute o aplicativo (AWS CLI)

Crie e execute 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`*

Edite a política do IAM

Edite a política do IAM para adicionar permissões para acessar o stream de dados do Kinesis e o stream do Firehose.

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 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. 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`

Execute 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.

Pare o aplicativo

Na MyApplicationpágina, escolha Parar. Confirme a ação.

Atualizar 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.

Crie e execute o aplicativo (AWS CLI)

Nesta seção, você usa o AWS CLI para criar e executar o aplicativo Managed Service for Apache Flink.

Criação de 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 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 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 “Criação de 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.

Crie o serviço gerenciado para o aplicativo 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 “Crie 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.

Inicie 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.

Pare 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 “Configurar o registro de aplicativos no Managed Service para Apache Flink”.

Atualizar o código do aplicativo

Quando precisar atualizar o código do aplicativo com uma nova versão do pacote de código, use a UpdateApplication AWS CLI ação.

Para usar o AWS CLI, exclua seu pacote de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo bucket e nome de objeto 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 “Crie 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 recursos da AWS

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial de introdução.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seu stream de dados do Kinesis
- Exclua seu stream do Firehose
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 stream 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 seu stream do Firehose

1. Abra o console do Kinesis em https://console.aws.amazon.com/kinesis.
2. No painel Firehose, escolha. ExampleDeliveryStream
3. Na ExampleDeliveryStreampágina, escolha Excluir stream do Firehose e confirme a exclusão.

Exclua 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 seu stream do Firehose, exclua esse bucket também.

Exclua seus recursos do IAM

1. Abra o console do 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. Se você criou uma nova política para seu stream do 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 seu stream do 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 de um stream do Kinesis em uma conta diferente

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

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 stream de origem do Kinesis
- Crie e atualize funções e políticas do IAM
- Atualize o script Python
- Atualizar o aplicativo Java

- Crie, carregue 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 Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink 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 stream do coletor. Use a AWS conta que você usou para o tutorial de introdução para o stream do aplicativo e do coletor. Use uma conta AWS diferente para o fluxo de origem.

Configuração

Você acessará suas duas AWS contas 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 AWS contas diferentes usando perfis.

Para obter mais informações sobre como usar perfis nomeados com o AWS CLI, consulte Perfis nomeados na AWS Command Line Interfacedocumentação.

Criar stream de origem 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 funções e políticas do IAM

Para permitir o acesso a objetos em todas AWS as contas, você cria uma função 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 Management Guia do usuário:

- Criando perfis do IAM
- Criando políticas do IAM

Funções e políticas da conta Sink

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
```

Funções 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 AWS Serviço. 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")
```

Atualizar 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");
    }
}
```

Crie, carregue 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 armazenamento confiável personalizado com o Amazon MSK

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

APIs de fonte de dados atuais

Se você estiver usando as APIs de fonte de dados atuais, seu aplicativo poderá aproveitar o utilitário Amazon 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. Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink

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 Kafka com segurança via TLS a um cluster Kafka, o cliente Kafka (como o exemplo do aplicativo Flink) deve confiar em toda a cadeia de confiança apresentada pelos certificados de servidor do cluster Kafka (da CA emissora até a CA de nível raiz). Como exemplo de um armazenamento confiável personalizado, usaremos um cluster Amazon MSK com a autenticação TLS mútua (MTLS) ativada. Isso significa que os nós do cluster MSK usam certificados de servidor emitidos por uma Autoridade de AWS Certificação Privada do Gerenciador de Certificados (CA Privada do ACM) que é privada de sua conta e região e, portanto, não é confiável para o armazenamento confiável padrão da Java Virtual Machine (JVM) que executa o aplicativo Flink.

> **Note**
>
> - Um armazenamento de chaves 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 armazenamento confiável é usado para armazenar certificados de Autoridades Certificadas (CA) 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 Apache Kafka personalizado hospedado em (Amazon AWSEC2 ou Amazon EKS)
- Um cluster 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 Amazon MSK
- Crie um armazenamento confiável personalizado e aplique-o ao seu cluster
- Crie o código do aplicativo
- Faça o upload do código Java de streaming do Apache Flink
- Criar o aplicativo
- Configurar o aplicativo
- Execute o aplicativo
- Teste o aplicativo

Crie uma VPC com um cluster 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 `AWS KafkaTutorialTopicDestination`:

```
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 os tutoriais de pré-requisitos, certifique-se de usar a AWS região selecionada no código, nos comandos e nas entradas do console.

Crie um armazenamento confiável 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 baixa e compila 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 de streaming do Apache Flink

Nesta seção, você faz o upload do código do seu aplicativo no bucket do Amazon S3 que você criou no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

> **Note**
>
> Se você excluiu o bucket do Amazon S3 no tutorial de introdução, siga a etapa the section called “Faça o upload do JAR arquivo de código do aplicativo” novamente.

1. No console do Amazon S3, escolha o ka-app-code- bucket e escolha Upload. <username>
2. Na etapa Selecionar arquivos, selecione 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	AWS KafkaTutorialTopic
`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	AWS KafkaTutorialTopic Destination
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 Use propriedades de tempo de execução no Managed Service para Apache Flink.

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.

Execute 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.

Teste 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 AWS KafkaTutorialTopicDestination --from-
beginning
```

Se nenhum registro aparecer no tópico de destino, consulte a seção Não é possível acessar recursos em um VPC no tópico Solucionar problemas de serviço gerenciado para Apache Flink.

Exemplos de Python

Os exemplos a seguir demonstram como criar aplicativos usando o Python com a API de tabelas do Apache Flink.

Tópicos

Exemplo: criação de uma janela giratória em Python

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

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 Tutorial: Comece a usar o Python no Managed Service para Apache Flink.

Este tópico contém as seguintes seções:

Crie 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>`**.

Grave 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 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 o upload do código Python de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie 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 Selecionar arquivos, selecione 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.

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

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**

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 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 Especifique 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. 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. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Edite 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. 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"
        }
    ]
}
```

Execute 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.

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Tumbling Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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: criação de uma janela deslizante em Python

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Tutorial: Comece a usar o Python no Managed Service para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Comprima e faça o upload do código Python de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Limpe AWS os recursos

Crie 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>`**.

Grave 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 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 o upload do código Python de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie 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 Selecionar arquivos, selecione 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.

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

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.
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 Especifique 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. 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. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Edite 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. 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"
        }
    ]
}
```

Execute 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.

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Sliding Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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 em Python

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

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 Tutorial: Comece a usar o Python no Managed Service para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave registros de amostra no fluxo de entrada
- Baixe e examine o código do aplicativo
- Comprima e faça o upload do código Python de streaming do Apache Flink
- Crie e execute o aplicativo Managed Service for Apache Flink
- Limpe AWS os recursos

Crie 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>`**.

Grave 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 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 o upload do código Python de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie 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 Selecionar arquivos, selecione 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.

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

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 Especifique 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`**

ID do grupo	Chave	Valor
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 Especifique 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. 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. Esse não é o mesmo fluxo de logs que o aplicativo usa para enviar resultados.

Edite 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. 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>/*"
            ]
        }
    ]
}
```

Execute 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.

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Sliding Window.

Este tópico contém as seguintes seções:

Exclua 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 stream 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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.

Exemplos de Scala

Os exemplos a seguir demonstram como criar aplicativos usando o Scala com o Apache Flink.

Tópicos

- Exemplo: criação de uma janela giratória no Scala
- Exemplo: criação de uma janela deslizante no Scala
- Exemplo: enviar dados de streaming para o Amazon S3 em Scala

Exemplo: criação de uma janela giratória no Scala

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

> **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
- Crie e execute o aplicativo (console)
- Crie e execute o aplicativo (CLI)
- Atualizar o código do aplicativo
- Limpe AWS os 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 Create bucket (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 Selecionar arquivos, selecione 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.

Crie e execute 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. Selecione 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`

Edite 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/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"
        }
    ]
}
```

Execute 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.

Pare o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa o AWS Command Line Interface para criar e executar o aplicativo Managed Service for Apache Flink. Use o AWS CLI comando kinesisanalyticsv2 para criar e interagir com o 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 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 AWS Serviço
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.

Inicie o aplicativo

Nesta seção, você usa a StartApplicationação 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.

Pare o aplicativo

Nesta seção, você usa a StopApplicationação 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.

Atualizar propriedades do ambiente

Nesta seção, você usa a UpdateApplicationação 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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 Crie 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"
                }
            }
        }
    }
}
```

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Tumbling Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis

- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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: criação de uma janela deslizante no Scala

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

> **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

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 Create bucket (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 Selecionar arquivos, selecione 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.

Crie e execute 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. Selecione 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`

Edite 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/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"
        }
    ]
}
```

Execute 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.

Pare o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa o AWS Command Line Interface para criar e executar o aplicativo Managed Service for Apache Flink. Use o AWS CLI comando kinesisanalyticsv2 para criar e interagir com o 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 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 AWS Serviço
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.

Inicie o aplicativo

Nesta seção, você usa a StartApplicationação 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.

Pare o aplicativo

Nesta seção, você usa a StopApplicationação 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.

Atualizar propriedades do ambiente

Nesta seção, você usa a UpdateApplicationação 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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 Crie 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"
                }
```

```
            }
        }
    }
}
```

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial da janela deslizante.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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 em Scala

> **Note**
>
> Para obter exemplos atuais, consulteExemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink.

> **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
- Crie e execute o aplicativo (console)
- Crie e execute o aplicativo (CLI)
- Atualizar o código do aplicativo
- Limpe AWS os 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 Create bucket (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 Selecionar arquivos, selecione 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.

Crie e execute 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. Selecione 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`

Edite 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: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"
        }
    ]
```

```
}
```

Execute 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.

Pare o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa o AWS Command Line Interface para criar e executar o aplicativo Managed Service for Apache Flink. Use o AWS CLI comando kinesisanalyticsv2 para criar e interagir com o 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 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 AWS Serviço
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.

Inicie o aplicativo

Nesta seção, você usa a StartApplicationação 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.

Pare o aplicativo

Nesta seção, você usa a StopApplicationação 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.

Atualizar propriedades do ambiente

Nesta seção, você usa a UpdateApplicationação 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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 Crie 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"
                }
            }
        }
    }
}
```

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Tumbling Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus recursos do IAM
- Exclua seus CloudWatch recursos

Exclua 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 streams 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.

Exclua 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.

Exclua seus recursos do IAM

1. Abra o console do 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 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.

Use um notebook Studio com serviço gerenciado para Apache Flink

Os notebooks Studio para Managed Service para Apache Flink permitem que você consulte interativamente fluxos de dados em tempo real e crie e execute facilmente aplicativos de processamento de fluxo usando padrões, SQL Python e Scala. Com alguns cliques no console AWS de gerenciamento, você pode iniciar um notebook 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: Criar um notebook Studio no Managed Service para Apache Flink Para obter mais informações sobre o Apache Zeppelin, consulte a documentação do Apache Zeppelin.

Com um notebook, você modela consultas usando o Apache Flink Table API & inSQL, SQL Python ou Scala, ou em Scala. DataStream API 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:

- Implemente como um aplicativo com estado durável
- Revise IAM as permissões dos notebooks Studio
- Use conectores e dependências
- Implemente funções definidas pelo usuário
- Ativar ponto de verificação
- Atualize o Studio Runtime
- Trabalhe com AWS Glue
- Exemplos e tutoriais para notebooks Studio no Managed Service for Apache Flink
- Solucionar problemas de notebooks Studio para serviço gerenciado para Apache Flink
- Crie IAM políticas personalizadas para o Managed Service para notebooks Apache Flink Studio

Use a versão correta do Studio Notebook Runtime

Com o Amazon Managed Service para Apache Flink Studio, você pode consultar fluxos de dados em tempo real e criar e executar aplicativos de processamento de streams usando padrões, SQL Python e Scala em um notebook interativo. Os notebooks Studio são equipados com o Apache Zeppelin e usam o Apache Flink como mecanismo de processamento de stream.

> **Note**
>
> Vamos descontinuar o Studio Runtime com o Apache Flink versão 1.11 em 5 de novembro de 2024. A partir dessa data, você não poderá executar novos notebooks ou criar novos aplicativos usando essa versão. Recomendamos que você atualize para o tempo de execução mais recente (Apache Flink 1.15 e Apache Zeppelin 0.10) antes disso. Para obter orientação sobre como atualizar seu notebook, consulteAtualize o Studio Runtime.

Tempo de execução do Studio

Versão Apache Flink	Versão Apache Zeppelin	Versão do Python	
1.15	0.10	3.8	Recomendado
1.13	0.9	3.8	Compatível

Versão Apache Flink	Versão Apache Zeppelin	Versão do Python	
1.11	0.9	3.7	Descontinuando em 5 de novembro de 2024

Crie um caderno 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 o 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 MSK cluster da Amazon, um stream de dados do Kinesis ou um bucket do Amazon S3.
- Um AWS Glue banco de dados. Esse banco de dados contém tabelas, que armazenam esquemas e endpoints de origem e destino de seus dados. Para obter mais informações, consulte Trabalhe com 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 AWS CLI o.

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

- No MSK console da Amazon, escolha seu cluster e escolha 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.

Execute uma análise interativa dos 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 SQL consulta a seguir mostra como recuperar dados de uma fonte de dados:

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

Para obter mais exemplos de SQL consultas do Flink Streaming, consulte a Exemplos e tutoriais para notebooks Studio no Managed Service for Apache Flink seguir e as consultas na documentação do Apache Flink.

Você pode usar SQL as consultas do Flink no notebook Studio para consultar dados de streaming. Você também pode usar Python (TabelaAPI) e Scala (Tabela e DatastreamAPIs) 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	Cria ExecutionEnvironment/ StreamExecutionEnvironment/BatchTableEnvironment/ StreamTableEnvironment e fornece um ambiente Scala
%flink.pyflink	PyFlinkInterpreter	Fornece um ambiente python
%flink.ipyflink	IPyFlinkInterpreter	Fornece um ambiente ipython
%flink.ssql	FlinkStreamSqlInterpreter	Fornece um ambiente stream sql
%flink.bsql	FlinkBatchSqlInterpreter	Fornece um ambiente sql em lote

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 for blink planner`

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

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

Para obter mais informações sobre o uso de ambientes de tabela, consulte Concepts and Common API na documentação do Apache Flink.

Implemente 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 SQL e/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 do AWS Management Console, AWS Command Line Interface ou AWS API alterar o paralelismo e/ou

ParallelismPer KPU as configurações 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, VPC deverá ter acesso à Internet. Se você VPC não tiver acesso à Internet, consulteImplemente como um aplicativo com estado durável sem acesso à Internet VPC.

Critérios Scala/Python

- Em seu código Scala ou Python, use o planejador Blink (`senv`, `stenv` para Scala`s_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 comandos IPython mágicos 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 a CASECLASSESdocumentação do Scala.

SQLcritérios

- SELECTDeclarações 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, DDL as declarações (`USE`,,`CREATE`,`ALTER`, `DROPSET`,`RESET`) devem preceder as declarações DML (`INSERT`). Isso ocorre porque DML as declarações em um parágrafo devem ser enviadas juntas como um único trabalho do Flink.
- Deve haver no máximo um parágrafo DML contendo declarações. 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 SQL funções com o Amazon Managed Service para Apache Flink, Amazon Translate e Amazon Comprehend.

Revise IAM as permissões dos notebooks Studio

O Managed Service for Apache Flink cria uma IAM função 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	Listar
Amazon EC2	Listar
AWS Glue	Ler, Gravar
Managed Service for Apache Flink	Ler
Managed Service for Apache Flink V2	Leitura
Amazon S3	Ler, Gravar

Use 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 conectores, consulte Tabela e SQL conectores 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-kinesis`, e. `flink-connector-kafka_2.12` `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 usar o CreateApplicationAPIpara criar seu notebook Studio, os `flink-connector-kafka` conectores `flink-sql-connector-flink` e não serão incluídos por padrão. Para adicioná-los,

especifique-os como a `MavenReference` no tipo de dados `CustomArtifactsConfiguration`, conforme mostrado nos exemplos a seguir.

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

Adicione 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, escolha a opção de 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 uma dependência JAR ou um conector personalizado ao criar um novo notebook Studio usando o CreateApplicationAPI, especifique a localização da JAR 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 UpdateApplicationAPIoperação e especifique a localização da JAR 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.

Implemente 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 JVM linguagem como Java ou Scala para implementar seus parágrafos UDFs em seu notebook Studio. Você também pode adicionar ao seu notebook Studio JAR arquivos externos que contenham UDFs implementados em uma JVM linguagem.

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

> **Note**
>
> Para ver um exemplo de configuração, consulte Traduzir, redigir e analisar dados de streaming usando SQL funções com o Amazon Managed Service para Apache Flink, Amazon Translate e Amazon Comprehend no blog do Machine Learning.AWS

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

1. Faça o upload UDF JAR do seu arquivo para o Amazon S3.
2. No AWS Management Console, escolha a opção de 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 JAR arquivo no Amazon S3 ou o ZIP arquivo que tem a implementação do seu. UDF
6. Escolha Salvar alterações.

Para adicionar um UDF JAR ao criar um novo notebook Studio usando o CreateApplicationAPI, especifique a JAR localização no tipo de `CustomArtifactConfiguration` dados. Para adicionar UDF JAR a a a um notebook Studio existente, invoque a UpdateApplicationAPIoperação e especifique a JAR localização no tipo de `CustomArtifactsConfigurationUpdate` dados. Como alternativa, você pode usar o AWS Management Console para adicionar UDF JAR arquivos ao seu notebook 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 explícito de createTemporarySystemfunção usando Function 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 de JAR em uma string minúscula.

  Por exemplo, se `TextAnalyticsUDF` for a classe principal para UDFJAR, um registro implícito resultará no nome `textanalyticsudf` da função. 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 você UDF JAR incluir o FlinkSDKs, configure seu projeto Java para que o UDF código-fonte possa ser compilado com o FlinkSDKs, mas as SDK classes do Flink não sejam incluídas no artefato de construção, por exemplo o. JAR

  Você pode usar `provided` scope no Apache Maven, declarações de `compileOnly` dependência no Gradle, `provided` scope in ou diretiva equivalente na configuração SBT de compilação do projeto. UDF Você pode consultar esse pom no exemplo do UDF frasco, que segue 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 SQL funções com o Amazon Managed Service para Apache Flink, Amazon Translate e Amazon Comprehend.

Ativar ponto de verificação

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.

Defina 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"
)
```

Defina 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"
)
```

Atualize o Studio Runtime

Esta seção contém informações sobre como atualizar o Studio Notebook Runtime. Recomendamos que você sempre atualize para o Studio Runtime compatível mais recente.

Atualize seu notebook para um novo Studio Runtime

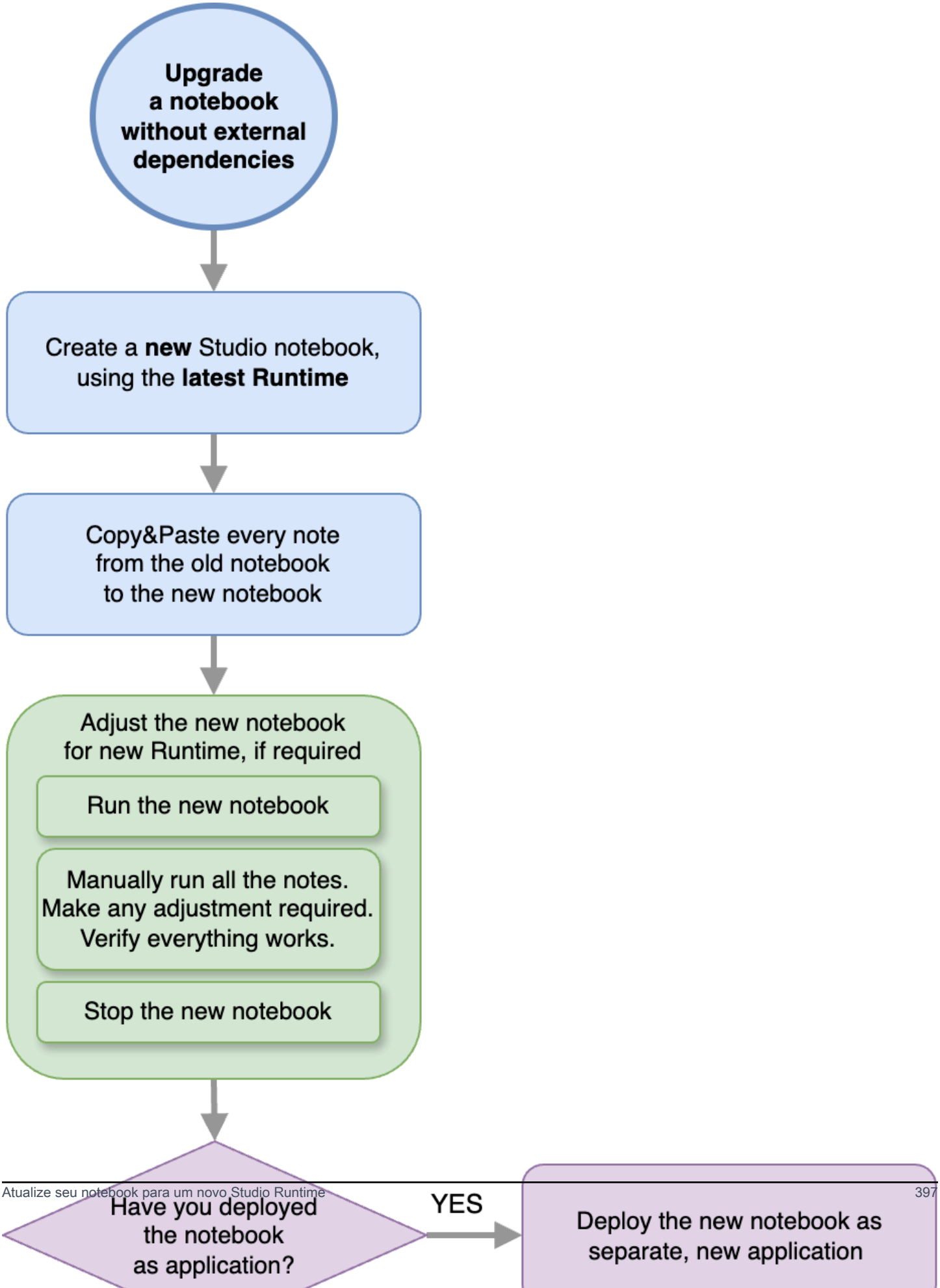
Dependendo de como você usa o Studio, as etapas para atualizar seu Runtime são diferentes. Selecione a opção adequada ao seu caso de uso.

SQLconsultas ou código Python sem dependências externas

Se você estiver usando SQL ou Python sem dependências externas, use o seguinte processo de atualização do Runtime. Recomendamos que você atualize para a versão mais recente do Runtime. O processo de atualização é o mesmo, independentemente da versão do Runtime da qual você está atualizando.

1. Crie um novo notebook Studio usando o Runtime mais recente.
2. Copie e cole o código de cada nota do caderno antigo para o novo.
3. No novo notebook, ajuste o código para torná-lo compatível com qualquer recurso do Apache Flink que tenha sido alterado em relação à versão anterior.
   - Execute o novo notebook. Abra o notebook e execute-o nota por nota, em sequência, e teste se funciona.
   - Faça as alterações necessárias no código.
   - Pare o novo caderno.
4. Se você tivesse implantado o notebook antigo como aplicativo:
   - Implante o novo notebook como um novo aplicativo separado.
   - Pare o aplicativo antigo.
   - Execute o novo aplicativo sem captura instantânea.
5. Pare o notebook antigo se ele estiver funcionando. Inicie o novo notebook, conforme necessário, para uso interativo.

Fluxo do processo para atualização sem dependências externas

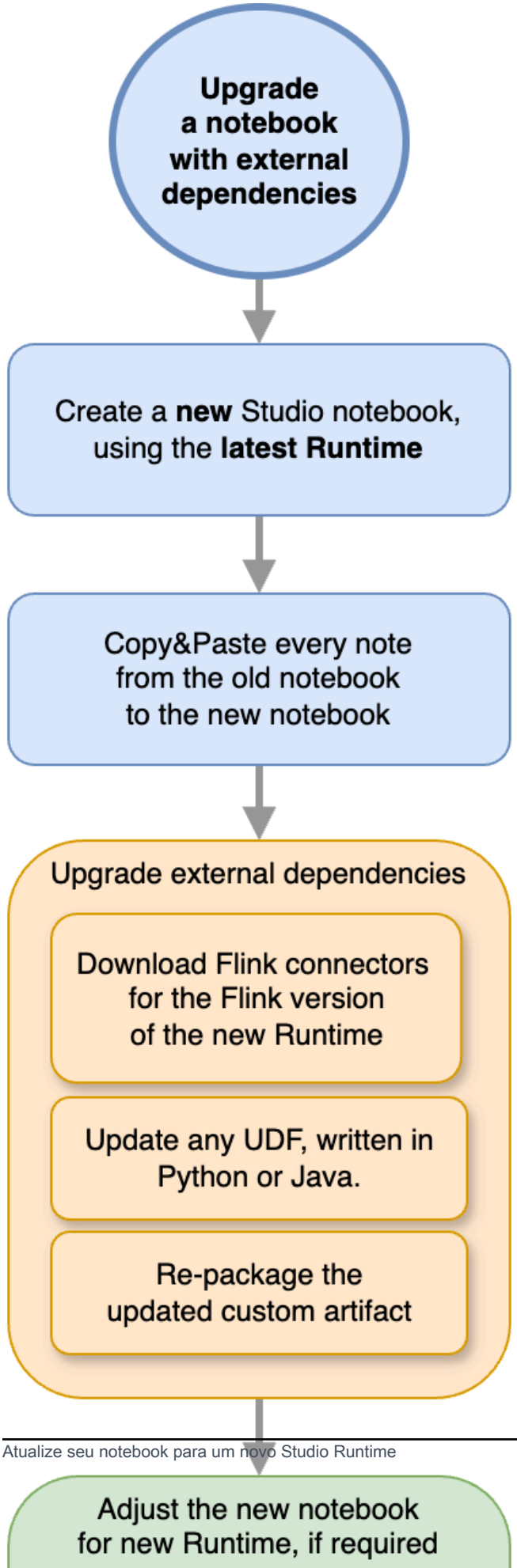

Upgrade a notebook without external dependencies
Create a new Studio notebook, using the latest Runtime
Copy&Paste every note from the old notebook to the new notebook
Adjust the new notebook for new Runtime, if required
Run the new notebook
Manually run all the notes. Make any adjustment required. Verify everything works.
Stop the new notebook
Have you deployed the notebook as application?
YES
Deploy the new notebook as separate, new application

SQLconsultas ou código Python com dependências externas

Siga esse processo se você estiver usando SQL ou Python e usando dependências externas, como conectores ou artefatos personalizados, como funções definidas pelo usuário implementadas em Python ou Java. Recomendamos que você atualize para o Runtime mais recente. O processo é o mesmo, independentemente da versão do Runtime da qual você está atualizando.

1. Crie um novo notebook Studio usando o Runtime mais recente.
2. Copie e cole o código de cada nota do caderno antigo para o novo.
3. Atualize as dependências externas e os artefatos personalizados.
   - Procure novos conectores compatíveis com a versão Apache Flink do novo Runtime. Consulte a Tabela e SQL os conectores na documentação do Apache Flink para encontrar os conectores corretos para a versão do Flink.
   - Atualize o código das funções definidas pelo usuário para corresponder às mudanças no Apache Flink API e em qualquer Python ou JAR dependências usadas pelas funções definidas pelo usuário. Empacote novamente seu artefato personalizado atualizado.
   - Adicione esses novos conectores e artefatos ao novo notebook.
4. No novo notebook, ajuste o código para torná-lo compatível com qualquer recurso do Apache Flink que tenha sido alterado em relação à versão anterior.
   - Execute o novo notebook. Abra o notebook e execute-o nota por nota, em sequência, e teste se funciona.
   - Faça as alterações necessárias no código.
   - Pare o novo caderno.
5. Se você tivesse implantado o notebook antigo como aplicativo:
   - Implante o novo notebook como um novo aplicativo separado.
   - Pare o aplicativo antigo.
   - Execute o novo aplicativo sem captura instantânea.
6. Pare o notebook antigo se ele estiver funcionando. Inicie o novo notebook, conforme necessário, para uso interativo.

Fluxo do processo para atualização com dependências externas


Upgrade a notebook with external dependencies
Create a new Studio notebook, using the latest Runtime
Copy&Paste every note from the old notebook to the new notebook
Upgrade external dependencies
Download Flink connectors for the Flink version of the new Runtime
Update any UDF, written in Python or Java.
Re-package the updated custom artifact
Adjust the new notebook for new Runtime, if required

Trabalhe com AWS Glue

Seu notebook Studio armazena e obtém informações sobre suas fontes de dados e AWS Glue coletores. Ao criar seu notebook Studio, você especifica o AWS Glue banco de dados que contém suas informações de conexão. Ao acessar suas fontes de dados e coletores, você especifica AWS Glue as tabelas 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 obter 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 Crie um AWS Glue banco de dados e Tutorial: Criar um notebook Studio no Managed Service para Apache Flink tutorial.

Propriedades da tabela

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

- Defina os 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.
- Use o conector Flink e as propriedades de formato: essas propriedades fornecem informações sobre seus fluxos de dados.

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

1. Faça login no AWS Management Console e abra o AWS Glue console 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.

Defina os 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 tempo de execução 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	O nome da coluna que AWS Glue será usada para expor o valor. Esse nome de coluna não corresponde a uma coluna de tabela existente.
Horário do evento	managed-flink.rowtime	O nome da coluna que AWS Glue será usada para expor o valor. Esse nome de coluna corresponde a uma coluna de tabela existente.
	managed-flink.watermark.*column_name*.milissegundos	O intervalo da marca d'água em milissegundos

Use 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`	O formato usado para desserializar e serializar

Tipo de conector	Chave	Valor
		mensagens do Kafka, por exemplo, ou. `json` `csv`
	`scan.startup.mode`	O modo de inicialização para o consumidor do Kafka, por exemplo, ou. `earliest-offset` `timestamp`
Kinesis	`format`	O formato usado para desserializar e serializar registros de stream de dados do Kinesis, por exemplo, ou. `json` `csv`
	`aws.region`	A AWS região em que o riacho é definido.
S3 (sistema de arquivos)	format	O formato usado para desserializar e serializar arquivos, por exemplo, ou. `json` `csv`
	`path`	O caminho do Amazon S3, por exemplo `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 para notebooks Studio no Managed Service for Apache Flink

Tópicos

- Tutorial: Criar um notebook Studio no Managed Service para Apache Flink

- Tutorial: Implantar um notebook Studio como um serviço gerenciado para o aplicativo Apache Flink com estado durável
- Veja exemplos de consultas para analisar dados em um notebook Studio

Tutorial: Criar um notebook Studio no Managed Service para Apache Flink

O tutorial a seguir demonstra como criar um notebook Studio que lê dados de um stream de dados do Kinesis ou de um cluster da AmazonMSK.

Este tutorial contém as seguintes seções:

- Conclua os pré-requisitos
- Crie um AWS Glue banco de dados
- Próximas etapas: Crie um notebook Studio com o Kinesis Data Streams ou a Amazon MSK
- Criar um bloco de anotações do Studio com o Kinesis Data Streams
- Crie um notebook Studio com a Amazon MSK
- Limpe seu aplicativo e os recursos dependentes

Conclua os pré-requisitos

Certifique-se de que sua versão AWS CLI seja 2 ou posterior. Para instalar a versão mais recente AWS CLI, consulte Instalação, atualização e desinstalação da AWS CLI versão 2.

Crie um AWS Glue banco de dados

Seu notebook Studio usa um AWS Gluebanco de dados para metadados sobre sua fonte de MSK dados da Amazon.

Criar um AWS Glue banco de dados

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

Próximas etapas: Crie um notebook Studio com o Kinesis Data Streams ou a Amazon MSK

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

- Criar um bloco de anotações do Studio com o 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.
- Crie um notebook Studio com a Amazon MSK: Com a AmazonMSK, você cria um aplicativo que usa um MSK cluster da Amazon como fonte. Você precisa criar uma AmazonVPC, uma instância EC2 cliente da Amazon e um MSK cluster da Amazon 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:

- Conclua os pré-requisitos
- Crie uma AWS Glue tabela
- 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

Conclua os pré-requisitos

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 sobre o 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 stream (`ExampleInputStream`) usando o 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
```

Crie uma AWS Glue tabela

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

Criar uma tabela

1. Faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. Se você ainda não tiver um AWS Glue banco de dados, escolha 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`** para o 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 a fonte do Kinesis URL, escolha enter. **`https://kinesis.us-east-1.amazonaws.com`** Se você copiar e colar a fonte do Kinesis URL, certifique-se de excluir todos os espaços à esquerda ou à direita. Escolha Próximo.
6. Na página Classificação, escolha JSON. Escolha 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
`ticker`	**`string`**
`price`	**`double`**

   Escolha 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 o AWS Management Console ou AWS CLI o.

- Crie um notebook Studio usando o AWS Management Console
- Crie um notebook Studio usando o AWS CLI

Crie um notebook Studio usando o AWS Management Console

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

> **Note**
>
> Você também pode criar um notebook Studio a partir dos consoles Amazon MSK ou Kinesis Data Streams selecionando seu cluster MSK Amazon de entrada ou stream de dados do Kinesis e escolhendo 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 o padrão para o banco de dados AWS 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 notebook Studio usando o AWS CLI

Para criar seu notebook Studio usando o 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 AWS CloudFormation console é aberto com o modelo Kinesis Data Generator. Escolha 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 isso pode criar recursos. AWS CloudFormation IAM caixa de seleção. Selecione Create Stack (Criar pilha).
7. Aguarde até que a AWS CloudFormation pilha termine de ser criada. Depois que a pilha estiver concluída, abra a pilha Kinesis-Data-Generator-Cognito-User no console e escolha a guia Saídas. AWS CloudFormation Abra a URL lista 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:

Region	**us-east-1**
Stream/Firehose stream	**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 serviço gerenciado para o console Apache Flink em casa https://console.aws.amazon.com/managed-flink/? region=us-east-1#/aplicativos/painel.
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, escolha. FLINKJOB 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 SQL consultas do Flink Streaming, consulte Consultas na documentação do Apache Flink.

Crie um notebook Studio com a Amazon MSK

Este tutorial descreve como criar um notebook Studio que usa um MSK cluster da Amazon como fonte.

Este tutorial contém as seguintes seções:

- Configurar um MSK cluster da Amazon
- Adicione um NAT gateway ao seu VPC
- Crie uma AWS Glue conexão e uma tabela
- Crie um notebook Studio com a Amazon MSK
- Envie dados para o seu MSK cluster da Amazon
- Teste seu bloco de anotações do Studio

Configurar um MSK cluster da Amazon

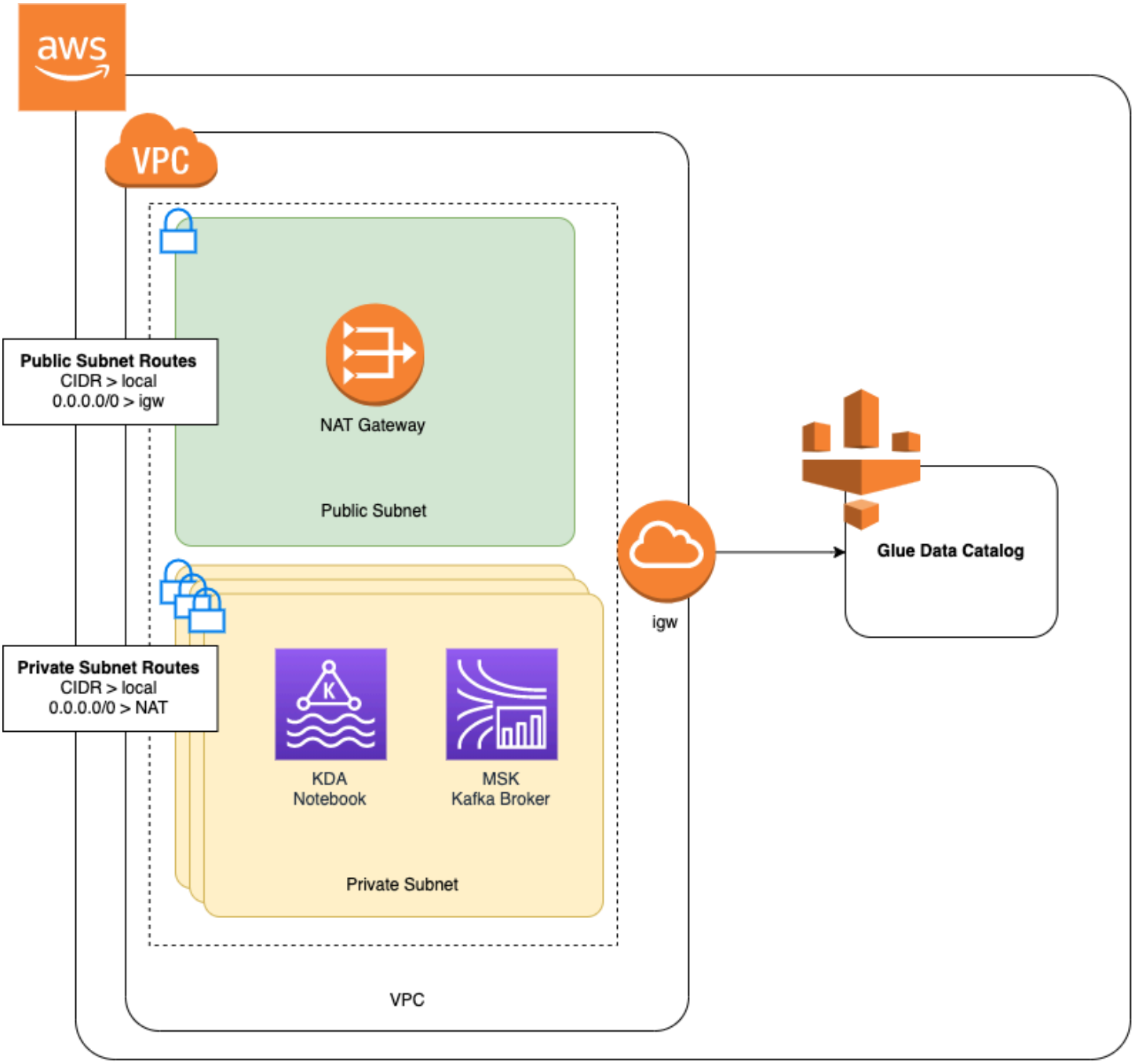
Para este tutorial, você precisa de um MSK cluster da Amazon que permita acesso em texto sem formatação. Se você ainda não tiver um MSK cluster da Amazon configurado, siga o MSK tutorial Como começar a usar a Amazon para criar uma AmazonVPC, um MSK cluster da Amazon, um tópico e uma instância EC2 cliente da Amazon.

Ao seguir o tutorial, faça o seguinte:

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

Adicione um NAT gateway ao seu VPC

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



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

1. Abra o VPC console da Amazon em https://console.aws.amazon.com/vpc/.
2. Escolha NATGateways na barra de navegação esquerda.
3. Na página NATGateways, escolha Create NAT Gateway.
4. Na página Create NAT Gateway, forneça os seguintes valores:

Nome - opcional	**`ZeppelinGateway`**
Sub-rede	AWS KafkaTutorialSubnet1
ID de alocação de IP elástico	Escolha um IP elástico disponível. Se não houver nenhum elástico IPs disponível, escolha Alocar IP elástico e, em seguida, escolha o IP elástico que o console cria.

Escolha Criar NAT gateway.

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 o seu VPC (por exemplo AWS KafkaTutorialVPC).

   Escolha Criar.
8. Na lista de tabelas de rotas, escolha 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 NATGateway, ZeppelinGateway. Selecione Salvar rotas. Escolha Fechar.
11. Na página Tabelas de rotas, com a ZeppelinRouteTableopção selecionada, escolha a guia Associações de sub-rede. Selecione Editar associações de sub-rede.
12. Na página Editar associações de sub-rede, escolha AWS KafkaTutorialSubnet2 e AWS KafkaTutorialSubnet3. Escolha Salvar.

Crie uma AWS Glue conexão e uma tabela

Seu notebook Studio usa um AWS Gluebanco de dados para metadados sobre sua fonte de MSK dados da Amazon. Nesta seção, você cria uma AWS Glue conexão que descreve como acessar seu

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

Criar uma conexão

1. Faça login no AWS Management Console e abra o AWS Glue console em https://console.aws.amazon.com/glue/.
2. Se você ainda não tiver um AWS Glue banco de dados, escolha 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 o servidor bootstrap Kafka URLs, forneça a string do broker bootstrap para seu cluster. Você pode obter os corretores de bootstrap no MSK console ou digitando o seguinte comando: CLI

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

   - Desmarque a caixa de seleção Exigir SSL conexão.

   Escolha Próximo.
5. Na VPCpágina, forneça os seguintes valores:

   - Para VPC, escolha o nome do seu VPC (por exemplo AWS KafkaTutorialVPC.)
   - Em Sub-rede, escolha AWS KafkaTutorialSubnet2.
   - 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 notebook no Apache Zeppelin para criar sua tabela por meio de uma instrução. DDL Em seguida, você pode fazer AWS Glue o check-in para verificar se 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 tópico (por exemplo AWS KafkaTutorialTopic). Para Conexão, escolha ZeppelinConnection.
4. Na página Classificação, escolha 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
`ticker`	**`string`**
`price`	**`double`**

   Escolha Próximo.
6. Na próxima página, verifique suas configurações e selecione Concluir.
7. Selecione a tabela recém-criada na lista de tabelas.
8. Escolha Editar tabela e adicione as seguintes propriedades:
   - chave:`managed-flink.proctime`, valor: `proctime`
   - chave:`flink.properties.group.id`, valor: `test-consumer-group`

- chave:`flink.properties.auto.offset.reset`, valor: `latest`
- chave:`classification`, valor: `json`

Sem esses pares de chave/valor, o notebook Flink apresenta um erro.

9. Selecione Apply (Aplicar).

Crie um notebook Studio com a 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 o AWS Management Console ou AWS CLI o.

- Crie um notebook Studio usando o AWS Management Console
- Crie um notebook Studio usando o AWS CLI

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

Crie um notebook Studio usando o AWS Management Console

1. Abra o serviço gerenciado para o console Apache Flink em casa https://console.aws.amazon.com/managed-flink/? region=us-east-1#/aplicativos/painel.
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 notebook Studio a partir dos consoles Amazon MSK ou Kinesis Data Streams, selecione seu cluster MSK Amazon de entrada ou stream de dados do Kinesis e escolha Processar dados em tempo real.

3. Na página Criar bloco de anotações do Studio, 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 MyNotebookpágina, escolha a guia Configuração. Na seção Redes, selecione Editar.
5. Na MyNotebook página Editar rede para, escolha a VPCconfiguração com base no MSK cluster da Amazon. Escolha seu MSK cluster da Amazon para o Amazon MSK Cluster. Escolha Salvar alterações.
6. Na MyNotebookpágina, escolha Executar. Aguarde até que o Status mostre Em execução.

Crie um notebook Studio usando o AWS CLI

Para criar seu notebook Studio usando o 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.
   - A sub-rede IDs e o ID do grupo de segurança da Amazon VPC que contém seu MSK cluster da Amazon.
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 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 seu MSK cluster da Amazon

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

1. Conecte-se ao seu EC2 cliente Amazon.
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 o AWS CLI em sua máquina cliente digitando o seguinte comando:

```
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 da amostra pela string Bootstrap Brokers do seu MSK cluster Amazon e atualize o nome do tópico se o tópico não for: AWS KafkaTutorialTopic

```
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 notebook Studio para consultar dados do seu MSK cluster da Amazon.

1. Abra o serviço gerenciado para o console Apache Flink em casa https://console.aws.amazon.com/managed-flink/? region=us-east-1#/aplicativos/painel.
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 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 MSK cluster da Amazon.

Para abrir o painel do Apache Flink para que seu aplicativo visualize aspectos operacionais, escolha. FLINKJOB 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 SQL consultas do Flink Streaming, consulte Consultas na documentação do Apache Flink.

Limpe seu aplicativo e os recursos dependentes

Exclua seu bloco de anotações do Studio

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

Exclua seu AWS Glue banco de dados e conexão

1. Abra o AWS Glue console 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 ZeppelinConnectionpara selecioná-la. Selecione Ação, Excluir conexão. Confirme a seleção.

Exclua sua IAM função e política

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

Excluir seu grupo CloudWatch de registros

O console cria um grupo de CloudWatch registros e um stream de registros 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 CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Selecione Grupos de log na barra de navegação à esquerda.
3. Escolha o grupo de MyNotebook registros AWSKinesisAnalytics///.
4. Selecione Actions (Ações), Delete log group(s) (Excluir grupo(s) de log). Confirme a exclusão.

Limpe 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.

Limpe MSK os recursos

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

Exclua seu MSK cluster da Amazon

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

1. Abra o MSK console da Amazon em https://console.aws.amazon.com/msk/casa? region=us-east-1#/home/.
2. Escolha AWS KafkaTutorialCluster. Escolha Excluir. Insira **`delete`** na janela que aparece e confirme sua seleção.

Encerrar sua instância cliente

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

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

Exclua sua Amazon VPC

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

1. Abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. Selecione Interfaces de rede na barra de navegação à esquerda.
3. Digite seu VPC ID 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 VPC console da Amazon em https://console.aws.amazon.com/vpc/.
9. Selecione AWS KafkaTutorialVPC. Escolha Ações, Excluir VPC. Entre **`delete`** e confirme a exclusão.

Tutorial: Implantar um notebook Studio como um serviço gerenciado para o aplicativo Apache Flink 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:

- Concluir os pré-requisitos
- Implemente um aplicativo com estado durável usando o AWS Management Console
- Implemente um aplicativo com estado durável usando o AWS CLI

Concluir os pré-requisitos

Crie um novo notebook Studio seguindo oTutorial: Criar um notebook Studio no Managed Service para Apache Flink, usando o Kinesis Data Streams MSK ou a Amazon. 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:

   - Amazon S3 FullAccess
   - Gerenciado pela Amazon - flinkFullAccess
   - Acesso às suas fontes, destinos e VPCs conforme aplicável. Para obter mais informações, consulte Revise IAM as permissões dos 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 escolha Implantar via AWS console.
- Isso levará você à AWS Management Console página de criação de um serviço gerenciado para o aplicativo Apache Flink. Observe que o nome do aplicativo, o paralelismo, a localização do código, o Glue DB padrão VPC (se aplicável) e as IAM funções foram pré-preenchidos. Valide se as IAM funções têm as permissões necessárias para suas fontes e destinos. Os snapshots são habilitados por padrão para um gerenciamento durável do estado do aplicativo.
- Selecione Create application (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 atualizá-lo 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 Conclua os pré-requisitos.

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 página do URL caderno Apache Zeppelin 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
```

Veja exemplos de consultas para analisar dados em um notebook Studio

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

- Crie tabelas com o MSK Amazon/Apache Kafka
- Crie tabelas com o Kinesis
- Consulte uma janela caindo
- Consulte uma janela deslizante
- Use interativo SQL
- Use o BlackHole SQL conector
- Use o Scala para gerar dados de amostra
- Use o Scala interativo
- Use Python interativo
- Use uma combinação de Python interativo e SQL Scala
- Use um stream de dados do Kinesis entre contas

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

Para visualizar seu aplicativo no painel do Apache Flink, escolha FLINKJOBna 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 do Apache Flink Streaming, consulte SQL Consultas na documentação do Apache Flink.

Crie tabelas com o MSK Amazon/Apache Kafka

Você pode usar o conector Amazon MSK Flink com o Managed Service para Apache Flink Studio para autenticar sua conexão com texto simples ou autenticação. SSL 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 Connector SQL.

Crie 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 Conector do Amazon Kinesis SQL Data Streams.

Consulte uma janela caindo

A seguinte SQL consulta do Flink Streaming seleciona o preço mais alto em cada janela de queda 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)
```

Consulte uma janela deslizante

A seguinte SQL consulta do Apache Flink Streaming 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)
```

Use interativo SQL

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 “Use o Scala para gerar dados de amostra”. Para testar outras SQL consultas, 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`;
```

Use o BlackHole SQL conector

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

Use o Scala para gerar dados de amostra

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`
```

Use o Scala interativo

Esta é a tradução em Scala do the section called “Use interativo SQL”. Para obter mais exemplos de Scala, consulte a Tabela API 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`
```

Use Python interativo

Esta é a tradução em Python do the section called “Use interativo SQL”. Para obter mais exemplos de Python, consulte a Tabela API 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`
```

Use uma combinação de Python interativo e SQL Scala

Você pode usar qualquer combinação de SQL Python e Scala em seu notebook para análise interativa. Em um notebook 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!")
```

Use um stream 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 DDL instrução create table para criar uma tabela em relação ao stream 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'
)
```

Solucionar problemas de notebooks Studio para serviço gerenciado para Apache Flink

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.

Implemente como um aplicativo com estado durável sem acesso à Internet VPC

A deploy-as-application função Managed Service for Apache Flink Studio não oferece suporte a VPC aplicativos 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

- Banco de dadosARN: *<glue database ARN (Amazon Resource Name)>*

5. Talvez seja necessário conceder permissões às IAM funções Managed Service for Apache Flink Studio e Managed Service for Apache Flink para os serviços que seu aplicativo usa. Você pode usar a mesma IAM função 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)
```

Reinicie 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.

Crie IAM políticas personalizadas para o Managed Service para notebooks Apache Flink Studio

Normalmente, você usa IAM políticas gerenciadas 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 IAM política personalizada. Esta seção contém exemplos de IAM políticas personalizadas.

> **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

O exemplo de política a seguir concede permissões para acessar um AWS Glue banco de dados.

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

MSKClusters da Amazon

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

Comece a usar o Amazon Managed Service para Apache Flink () DataStream API

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink e a implementação de um aplicativo em Java usando o. DataStream API 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

Analise os componentes do aplicativo Managed Service for Apache Flink

> **Note**
>
> O Amazon Managed Service para Apache Flink oferece suporte a todo o Apache Flink APIs e, potencialmente, a todos os idiomas. JVM Para obter mais informações, consulte Flink's APIs.
> Dependendo da API sua escolha, a estrutura do aplicativo e a implementação são um pouco diferentes. Este tutorial de introdução aborda a implementação dos aplicativos usando o DataStream API em Java.

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java que processa a entrada e produz a saída usando o tempo de execução do Apache Flink.

Um aplicativo típico do Managed Service for Apache Flink tem os seguintes componentes:

- Propriedades de tempo de execução: você pode usar propriedades de tempo de execução para passar parâmetros de configuração para seu aplicativo e alterá-los sem modificar e republicar o código.
- Fontes: o aplicativo consome dados de uma ou mais fontes. Uma fonte usa um conector para ler dados de um sistema externo, como um stream de dados do Kinesis ou um bucket do Kafka. Para obter mais informações, consulte Adicione fontes de dados de streaming ao Managed Service for Apache Flink.
- Operadores: o aplicativo processa dados usando um ou mais operadores. Um operador pode transformar, enriquecer ou agregar dados. Para obter mais informações, consulte Transforme dados usando operadores no Managed Service for Apache Flink.
- Coletores: o aplicativo envia dados para fontes externas por meio de coletores. Um coletor usa um conector v para enviar dados para um stream de dados do Kinesis, um tópico do Kafka, Amazon S3 ou um banco de dados relacional. Você também pode usar um conector especial para imprimir a saída somente para fins de desenvolvimento. Para obter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

Seu aplicativo requer algumas dependências externas, como os conectores Flink que seu aplicativo usa ou, potencialmente, uma biblioteca Java. Para ser executado no Amazon Managed Service para Apache Flink, o aplicativo deve ser empacotado junto com as dependências em um fat-jar e carregado em um bucket do Amazon S3. Em seguida, crie um aplicativo do Managed Service for Apache Flink. Você passa a localização do pacote de código, junto com qualquer outro parâmetro de configuração de tempo de execução.

Este tutorial demonstra como usar o Apache Maven para empacotar o aplicativo e como executá-lo localmente no local IDE de sua escolha.

Cumpra os pré-requisitos para concluir os exercícios

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

- Cliente do Git. Instale o cliente Git, caso ainda não tenha feito isso.
- Java Development Kit (JDK) versão 11. Instale um Java JDK 11 e defina a variável de `JAVA_HOME` ambiente para apontar para seu local de JDK instalação. Se você não tiver um JDK 11, poderá usar o Amazon Coretto 11 ou qualquer outro padrão JDK de sua escolha.

- Para verificar se você o JDK instalou corretamente, execute o comando a seguir. A saída será diferente se você estiver usando um que não JDK seja o Amazon Corretto. Verifique se a versão é 11.x.

```
$ java --version

openjdk 11.0.23 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-11.0.23.9.1 (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.23.9.1 (build 11.0.23+9-LTS, mixed mode)
```

- Apache Maven. Instale o Apache Maven, caso ainda não tenha feito isso. Para saber como instalá-lo, consulte Instalando o Apache Maven.
  - Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

- IDEpara o desenvolvimento local. Recomendamos que você use um ambiente de desenvolvimento, como Eclipse, Java Neon ou IntelliJ, para desenvolver e IDEAcompilar seu aplicativo.
  - Para testar a instalação do Apache Maven, insira o seguinte:

```
$ mvn -version
```

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

Configurar uma AWS conta e criar um usuário administrador

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

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

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

Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira 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 seu usuário root.

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

Criar um usuário com acesso administrativo

1. Ative o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No IAM Identity Center, conceda acesso administrativo a um usuário.

Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para entrar com seu usuário do IAM Identity Center, use o login URL que foi enviado ao seu endereço de e-mail quando você criou o usuário do IAM Identity Center.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No IAM Identity Center, crie um conjunto de permissões que siga as melhores práticas de aplicação de permissões com privilégios mínimos.

   Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .
2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando 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 o AWS	Siga as instruções da interface que deseja utilizar.

Qual usuário precisa de acesso programático?	Para	Por
(Usuários gerenciados no IAM Identity Center)	CLI AWS SDKs, ou. AWS APIs	• Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte Autenticação do IAM Identity Center no Guia de referência de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções em Uso de credenciais temporárias com AWS recursos no Guia do IAM usuário.

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

Próxima etapa

Configure o AWS Command Line Interface (AWS CLI)

Configure o AWS Command Line Interface (AWS CLI)

Nesta etapa, você baixa e configura o AWS CLI para usar com o Managed Service para 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 tem o AWS CLI instalado, talvez seja necessário fazer o upgrade para obter a funcionalidade mais recente. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface . Para verificar a versão do AWS CLI, execute o seguinte comando:
>
> ```
> aws --version
> ```
>
> Os exercícios deste tutorial exigem a seguinte AWS CLI versão ou posterior:
>
> ```
> aws-cli/1.16.63
> ```

Para configurar o 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 AWS CLI `config` arquivo. 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 AWS regiões 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 us-east-1 Leste dos EUA (Norte da Virgínia). 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 a AWS CLI, você pode tentar o próximo exercício, no qual você configura um aplicativo de amostra e testa a end-to-end configuração.

Próxima etapa

Crie e execute um serviço gerenciado para o aplicativo Apache Flink

Crie e execute um serviço gerenciado para o aplicativo Apache Flink

Nesta etapa, você cria um serviço gerenciado para o aplicativo Apache Flink com fluxos de dados do Kinesis como fonte e coletor.

Esta seção contém as seguintes etapas:

- Crie recursos dependentes
- Configurar seu ambiente de desenvolvimento local
- Baixe e examine o código Java de streaming do Apache Flink
- Grave registros de amostra no fluxo de entrada
- Execute seu aplicativo localmente
- Observe os dados de entrada e saída nos streams do Kinesis
- Pare de executar seu aplicativo localmente
- Compile e empacote o código do seu aplicativo

Crie recursos dependentes

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Dois streams de dados do Kinesis para entrada e saída
- Um bucket Amazon S3 para armazenar o código do aplicativo

> **Note**
>
> Este tutorial pressupõe que você esteja implantando seu aplicativo na região us-east-1 Leste dos EUA (Norte da Virgínia). Se você usa outra região, adapte todas as etapas adequadamente.

Crie dois streams de dados do Amazon 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 streams usando o console do Amazon Kinesis ou o comando a AWS CLI seguir. Para obter instruções sobre o console, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams. Para criar os fluxos usando o AWS CLI, use os comandos a seguir, ajustando-se à região que você usa para seu aplicativo.

Como criar os fluxos de dados (AWS CLI)

1. Para criar o primeiro stream (`ExampleInputStream`), 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 \
```

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

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

Crie um bucket Amazon S3 para o código do aplicativo

Você pode criar um bucket do Amazon S3 usando o console. Para saber como criar um bucket do Amazon S3 usando o console, consulte Criação de um bucket no Guia do usuário do Amazon S3. Nomeie o bucket do Amazon S3 usando um nome globalmente exclusivo, por exemplo, anexando seu nome de login.

> **Note**
>
> Certifique-se de criar o bucket na região que você usa para este tutorial (us-east-1).

Outros recursos

Quando você cria seu aplicativo, o Managed Service for Apache Flink cria automaticamente os seguintes CloudWatch recursos da Amazon, caso eles ainda não existam:

- Um grupo de logs chamado `/AWS/KinesisAnalytics-java/<my-application>`
- Um fluxo de logs chamado `kinesis-analytics-log-stream`

Configurar seu ambiente de desenvolvimento local

Para desenvolvimento e depuração, você pode executar o aplicativo Apache Flink em sua máquina diretamente da sua escolha. IDE Todas as dependências do Apache Flink são tratadas como dependências Java regulares usando o Apache Maven.

> **Note**
>
> Em sua máquina de desenvolvimento, você deve ter o Java JDK 11, o Maven e o Git instalados. Recomendamos que você use um ambiente de desenvolvimento como o Eclipse, Java Neon ou IntelliJ. IDEA Para verificar se você atende a todos os pré-requisitos, consulte. Cumpra os pré-requisitos para concluir os exercícios Você não precisa instalar um cluster Apache Flink em sua máquina.

Autentique sua sessão AWS

O aplicativo usa fluxos de dados do Kinesis para publicar dados. Ao executar localmente, você deve ter uma sessão AWS autenticada válida com permissões para gravar no stream de dados do Kinesis. Use as etapas a seguir para autenticar sua sessão:

1. Se você não tiver o AWS CLI e um perfil nomeado com credencial válida configurado, consulteConfigure o AWS Command Line Interface (AWS CLI).
2. Verifique se o seu AWS CLI está configurado corretamente e se seus usuários têm permissões para gravar no stream de dados do Kinesis publicando o seguinte registro de teste:

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. Se você IDE tiver um plug-in com o qual se integrar AWS, poderá usá-lo para passar as credenciais para o aplicativo em execução noIDE. Para obter mais informações, consulte AWS Toolkit for IDEA AWS IntelliJ e Toolkit for Eclipse.

Baixe e examine o código Java de streaming do Apache Flink

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-managed-service-for-apache-flink-examples.git
```

2. Navegue até o diretório `amazon-managed-service-for-apache-flink-examples/tree/main/java/GettingStarted`.

Analise os componentes do aplicativo

O aplicativo é totalmente implementado na `com.amazonaws.services.msf.BasicStreamingJob` classe. O `main()` método define o fluxo de dados para processar os dados de streaming e executá-los.

> **Note**
>
> Para uma experiência de desenvolvedor otimizada, o aplicativo foi projetado para ser executado sem nenhuma alteração de código no Amazon Managed Service para Apache Flink e localmente, para desenvolvimento em seu. IDE

- Para ler a configuração de tempo de execução para que ela funcione ao ser executada no Amazon Managed Service para Apache Flink e no seuIDE, o aplicativo detecta automaticamente se está sendo executado de forma independente localmente no. IDE Nesse caso, o aplicativo carrega a configuração do tempo de execução de forma diferente:
  1. Quando o aplicativo detectar que está sendo executado no modo autônomo no seuIDE, forme o `application_properties.json` arquivo incluído na pasta de recursos do projeto. O conteúdo do arquivo segue.
  2. Quando o aplicativo é executado no Amazon Managed Service para Apache Flink, o comportamento padrão carrega a configuração do aplicativo a partir das propriedades de tempo de execução que você definirá no aplicativo Amazon Managed Service para Apache Flink. Consulte Crie e configure o serviço gerenciado para o aplicativo Apache Flink.

```
private static Map<String, Properties>
 loadApplicationProperties(StreamExecutionEnvironment env) throws IOException {
    if (env instanceof LocalStreamEnvironment) {
        LOGGER.info("Loading application properties from '{}'",
 LOCAL_APPLICATION_PROPERTIES_RESOURCE);
        return KinesisAnalyticsRuntime.getApplicationProperties(
                BasicStreamingJob.class.getClassLoader()

 .getResource(LOCAL_APPLICATION_PROPERTIES_RESOURCE).getPath());
    } else {
```

```
        LOGGER.info("Loading application properties from Amazon Managed Service for
 Apache Flink");
        return KinesisAnalyticsRuntime.getApplicationProperties();
    }
}
```

- O `main()` método define o fluxo de dados do aplicativo e o executa.
  - Inicializa os ambientes de streaming padrão. Neste exemplo, mostramos como criar o `StreamExecutionEnvironment` a ser usado com o DataSteam API e o `StreamTableEnvironment` a ser usado com SQL e a TabelaAPI. Os dois objetos de ambiente são duas referências separadas ao mesmo ambiente de tempo de execução, para uso diferenteAPIs.

```
StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();
```

  - Carregue os parâmetros de configuração do aplicativo. Isso os carregará automaticamente do local correto, dependendo de onde o aplicativo está sendo executado:

```
Map<String, Properties> applicationParameters = loadApplicationProperties(env);
```

  - O aplicativo define uma fonte usando o conector Kinesis Consumer para ler dados do stream de entrada. A configuração do fluxo de entrada é definida no `PropertyGroupId =InputStream0`. O nome e a região do fluxo estão nas propriedades nomeadas `stream.name` e`aws.region`, respectivamente. Para simplificar, essa fonte lê os registros como uma string.

```
private static FlinkKinesisConsumer<String> createSource(Properties
 inputProperties) {
    String inputStreamName = inputProperties.getProperty("stream.name");
    return new FlinkKinesisConsumer<>(inputStreamName, new SimpleStringSchema(),
 inputProperties);
}
...

public static void main(String[] args) throws Exception {
   ...
   SourceFunction<String> source =
 createSource(applicationParameters.get("InputStream0"));
   DataStream<String> input = env.addSource(source, "Kinesis Source");
   ...
```

```
}
```

- Em seguida, o aplicativo define um coletor usando o conector Kinesis Streams Sink para enviar dados para o stream de saída. O nome e a região do fluxo de saída são definidos em `PropertyGroupId =OutputStream0`, semelhante ao fluxo de entrada. O coletor é conectado diretamente ao interno `DataStream` que está recebendo dados da fonte. Em um aplicativo real, você tem alguma transformação entre fonte e coletor.

```
private static KinesisStreamsSink<String> createSink(Properties outputProperties) {
    String outputStreamName = outputProperties.getProperty("stream.name");
    return KinesisStreamsSink.<String>builder()
            .setKinesisClientProperties(outputProperties)
            .setSerializationSchema(new SimpleStringSchema())
            .setStreamName(outputStreamName)
            .setPartitionKeyGenerator(element ->
 String.valueOf(element.hashCode()))
            .build();
}
...
public static void main(String[] args) throws Exception {
   ...
   Sink<String> sink = createSink(applicationParameters.get("OutputStream0"));
   input.sinkTo(sink);
   ...
}
```

- Por fim, você executa o fluxo de dados que acabou de definir. Essa deve ser a última instrução do `main()` método, depois de definir todos os operadores que o fluxo de dados exige:

```
env.execute("Flink streaming Java API skeleton");
```

Use o arquivo pom.xml

O arquivo pom.xml define todas as dependências exigidas pelo aplicativo e configura o plug-in Maven Shade para criar o fat-jar que contém todas as dependências exigidas pelo Flink.

- Algumas dependências têm `provided` escopo. Essas dependências estão disponíveis automaticamente quando o aplicativo é executado no Amazon Managed Service para Apache Flink. Eles são necessários para compilar o aplicativo ou para executá-lo localmente em seuIDE. Para obter mais informações, consulte Execute seu aplicativo localmente. Certifique-se de usar a

mesma versão do Flink do tempo de execução que você usará no Amazon Managed Service para Apache Flink.

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-clients</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
```

- Você deve adicionar dependências adicionais do Apache Flink ao pom com o escopo padrão, como o conector Kinesis usado por esse aplicativo. Para obter mais informações, consulte Use conectores Apache Flink com o Managed Service para Apache Flink. Você também pode adicionar quaisquer dependências Java adicionais exigidas pelo seu aplicativo.

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-kinesis</artifactId>
    <version>${aws.connector.version}</version>
</dependency>
```

- O plug-in Maven Java Compiler garante que o código seja compilado em Java 11, a JDK versão atualmente suportada pelo Apache Flink.
- O plug-in Maven Shade empacota o fat-jar, excluindo algumas bibliotecas que são fornecidas pelo tempo de execução. Também especifica dois transformadores: e. `ServicesResourceTransformer` `ManifestResourceTransformer` O último configura a classe que contém o `main` método para iniciar o aplicativo. Se você renomear a classe principal, não se esqueça de atualizar esse transformador.
-

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
```

```
    ...
        <transformer
 implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
            <mainClass>com.amazonaws.services.msf.BasicStreamingJob</mainClass>
        </transformer>
    ...
</plugin>
```

Grave registros de amostra no fluxo de entrada

Nesta seção, você enviará registros de amostra ao stream para o aplicativo processar. Você tem duas opções para gerar dados de amostra, usando um script Python ou o Kinesis Data Generator.

Gere dados de amostra usando um script Python

Você pode usar um script Python para enviar registros de amostra para o stream.

> **Note**
>
> Para executar esse script Python, você deve usar o Python 3.x e ter a biblioteca for AWS SDKPython (Boto) instalada.

Para começar a enviar dados de teste para o stream de entrada do Kinesis:

1. Baixe o script `stock.py` Python do gerador de dados no repositório do gerador GitHub de dados.
2. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o resto do tutorial. Agora você pode executar seu aplicativo Apache Flink.

Gere dados de amostra usando o Kinesis Data Generator

Como alternativa ao script Python, você pode usar o Kinesis Data Generator, também disponível em uma versão hospedada, para enviar dados de amostra aleatórios para o stream. O Kinesis Data Generator é executado no seu navegador e você não precisa instalar nada na sua máquina.

Para configurar e executar o Kinesis Data Generator:

1. Siga as instruções na documentação do Kinesis Data Generator para configurar o acesso à ferramenta. Você executará um AWS CloudFormation modelo que configura um usuário e uma senha.
2. Acesse o Kinesis Data Generator por meio do URL gerado pelo CloudFormation modelo. Você pode encontrar o URL na guia Saída após a conclusão do CloudFormation modelo.
3. Configure o gerador de dados:
   - Região: selecione a região que você está usando para este tutorial: us-east-1
   - Stream/stream de entrega: selecione o stream de entrada que o aplicativo usará: `ExampleInputStream`
   - Registros por segundo: 100
   - Modelo de registro: copie e cole o seguinte modelo:

```
{
  "event_time" : "{{date.now("YYYY-MM-DDTkk:mm:ss.SSSSS")}},
  "ticker" : "{{random.arrayElement(
        ["AAPL", "AMZN", "MSFT", "INTC", "TBV"]
    )}}",
  "price" : {{random.number(100)}}
}
```

4. Teste o modelo: escolha Modelo de teste e verifique se o registro gerado é semelhante ao seguinte:

```
{ "event_time" : "2024-06-12T15:08:32.04800, "ticker" : "INTC", "price" : 7 }
```

5. Inicie o gerador de dados: escolha Selecionar Enviar dados.

O Kinesis Data Generator agora está enviando dados para o. `ExampleInputStream`

Execute seu aplicativo localmente

Você pode executar e depurar seu aplicativo Flink localmente em seu. IDE

> **Note**
>
> Antes de continuar, verifique se os fluxos de entrada e saída estão disponíveis. Consulte Crie dois streams de dados do Amazon Kinesis. Além disso, verifique se você tem permissão para ler e gravar nos dois fluxos. Consulte Autentique sua sessão AWS.
> A configuração do ambiente de desenvolvimento local requer Java 11JDK, Apache Maven e e IDE para desenvolvimento em Java. Verifique se você atende aos pré-requisitos exigidos. Consulte Cumpra os pré-requisitos para concluir os exercícios.

Importe o projeto Java para o seu IDE

Para começar a trabalhar no aplicativo em seuIDE, você deve importá-lo como um projeto Java.

O repositório que você clonou contém vários exemplos. Cada exemplo é um projeto separado. Para este tutorial, importe o conteúdo do `./java/GettingStarted` subdiretório para o seuIDE.

Insira o código como um projeto Java existente usando o Maven.

> **Note**
>
> O processo exato para importar um novo projeto Java varia de acordo com o IDE que você está usando.

Verifique a configuração do aplicativo local

Ao ser executado localmente, o aplicativo usa a configuração no `application_properties.json` arquivo na pasta de recursos do projeto abaixo`./src/main/resources`. Você pode editar esse arquivo para usar diferentes nomes ou regiões de stream do Kinesis.

```
[
  {
    "PropertyGroupId": "InputStream0",
    "PropertyMap": {
      "stream.name": "ExampleInputStream",
      "flink.stream.initpos": "LATEST",
      "aws.region": "us-east-1"
```

```
    }
  },
  {
    "PropertyGroupId": "OutputStream0",
    "PropertyMap": {
      "stream.name": "ExampleOutputStream",
      "aws.region": "us-east-1"
    }
  }
]
```

Defina sua configuração de IDE execução

Você pode executar e depurar o aplicativo Flink IDE diretamente de você executando a classe principal`com.amazonaws.services.msf.BasicStreamingJob`, da mesma forma que executaria qualquer aplicativo Java. Antes de executar o aplicativo, você deve definir a configuração Executar. A configuração depende do IDE que você está usando. Por exemplo, consulte Configurações de execução/depuração na documentação do IntelliJ. IDEA Em particular, você deve configurar o seguinte:

1. Adicione as **`provided`** dependências ao classpath. Isso é necessário para garantir que as dependências com `provided` escopo sejam passadas para o aplicativo quando executado localmente. Sem essa configuração, o aplicativo exibe um `class not found` erro imediatamente.
2. Passe as AWS credenciais para acessar os streams do Kinesis para o aplicativo. A maneira mais rápida é usar o AWS Toolkit for IDEA IntelliJ. Usando esse IDE plug-in na configuração Executar, você pode selecionar um AWS perfil específico. AWS a autenticação acontece usando esse perfil. Você não precisa passar as AWS credenciais diretamente.
3. Verifique se ele IDE executa o aplicativo usando JDK11.

Execute o aplicativo em seu IDE

Depois de definir a configuração Executar para o`BasicStreamingJob`, você pode executá-la ou depurá-la como um aplicativo Java comum.

> **Note**
>
> Você não pode executar o fat-jar gerado pelo Maven diretamente na linha `java -jar ...` de comando. Esse jar não contém as dependências principais do Flink necessárias para executar o aplicativo de forma independente.

Quando o aplicativo é iniciado com êxito, ele registra algumas informações sobre o minicluster autônomo e a inicialização dos conectores. Isso é seguido por vários INFO e alguns WARN registros que o Flink normalmente emite quando o aplicativo é iniciado.

```
13:43:31,405 INFO  com.amazonaws.services.msf.BasicStreamingJob                [] -
 Loading application properties from 'flink-application-properties-dev.json'
13:43:31,549 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Flink Kinesis Consumer is going to read the following streams:
 ExampleInputStream,
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.cpu.cores required for local execution is not
 set, setting it to the maximal possible value.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils
 [] - The configuration option taskmanager.memory.task.heap.size required for local
 execution is not set, setting it to the maximal possible value.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.memory.task.off-heap.size required for local
 execution is not set, setting it to the maximal possible value.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.memory.network.min required for local execution
 is not set, setting it to its default value 64 mb.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.memory.network.max required for local execution
 is not set, setting it to its default value 64 mb.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils [] -
 The configuration option taskmanager.memory.managed.size required for local execution
 is not set, setting it to its default value 128 mb.
13:43:31,677 INFO  org.apache.flink.runtime.minicluster.MiniCluster             [] -
 Starting Flink Mini Cluster
....
```

Depois que a inicialização for concluída, o aplicativo não emitirá mais nenhuma entrada de registro. Enquanto os dados estão fluindo, nenhum registro é emitido.

Para verificar se o aplicativo está processando dados corretamente, você pode inspecionar os streams de entrada e saída do Kinesis, conforme descrito na seção a seguir.

> **Note**
>
> Não emitir registros sobre o fluxo de dados é o comportamento normal de um aplicativo Flink. A emissão de registros em cada registro pode ser conveniente para depuração, mas pode adicionar uma sobrecarga considerável durante a execução em produção.

Observe os dados de entrada e saída nos streams do Kinesis

Você pode observar os registros enviados ao stream de entrada pelo (gerando o Python de amostra) ou pelo Kinesis Data Generator (link) usando o Visualizador de dados no console do Amazon Kinesis.

Para observar registros

1. Abra o console do Kinesis em https://console.aws.amazon.com /kinesis.
2. Verifique se a região é a mesma em que você está executando este tutorial, que é us-east-1 Leste dos EUA (Norte da Virgínia) por padrão. Altere a região se ela não corresponder.
3. Escolha fluxos de dados.
4. Selecione o fluxo que você deseja observar, `ExampleInputStream` ou `ExampleOutputStream`.
5. Escolha a guia Visualizador de dados.
6. Escolha qualquer fragmento, mantenha Último como posição inicial e escolha Obter registros. Talvez você veja o erro “Nenhum registro encontrado para esta solicitação”. Em caso afirmativo, escolha Tentar obter registros novamente. Os registros mais recentes publicados no stream são exibidos.
7. Escolha o valor na coluna Dados para inspecionar o conteúdo do registro em JSON formato.

Pare de executar seu aplicativo localmente

Pare a execução do aplicativo em seuIDE. IDEGeralmente fornece uma opção de “parar”. A localização e o método exatos dependem do IDE que você está usando.

Compile e empacote o código do seu aplicativo

Nesta seção, você usa o Apache Maven para compilar o código Java e empacotá-lo em um arquivo. JAR Você pode compilar e empacotar seu código usando a ferramenta de linha de comando Maven ou seu. IDE

Para compilar e empacotar usando a linha de comando do Maven:

Vá para o diretório que contém o GettingStarted projeto Java e execute o seguinte comando:

```
$ mvn package
```

Para compilar e empacotar usando seuIDE:

Execute `mvn package` a partir da sua integração com o IDE Maven.

Em ambos os casos, o seguinte JAR arquivo é criado:`target/amazon-msf-java-stream-app-1.0.jar`.

> **Note**
>
> Executar um “projeto de compilação” a partir do seu IDE pode não criar o JAR arquivo.

Faça o upload do JAR arquivo de código do aplicativo

Nesta seção, você carrega o JAR arquivo criado na seção anterior para o bucket do Amazon Simple Storage Service (Amazon S3) que você criou no início deste tutorial. Se você não concluiu essa etapa, consulte (link).

Para carregar o JAR arquivo de código do aplicativo

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Escolha o bucket que você criou anteriormente para o código do aplicativo.
3. Escolha Carregar.
4. Escolha Adicionar arquivos.
5. Navegue até o JAR arquivo gerado na etapa anterior:`target/amazon-msf-java-stream-app-1.0.jar`.
6. Escolha Carregar sem alterar nenhuma outra configuração.

> **Warning**
>
> Certifique-se de selecionar o JAR arquivo correto em`<repo-dir>/java/GettingStarted/target/amazon-msf-java-stream-app-1.0.jar`.
> O `target` diretório também contém outros JAR arquivos que você não precisa carregar.

Crie e configure o serviço gerenciado para o aplicativo Apache Flink

Você pode criar e executar um aplicativo Managed Service for Apache Flink usando o console ou a AWS CLI. Para este tutorial, você usará o console.

> **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. Ao criar o aplicativo usando o AWS CLI, você cria esses recursos separadamente.

Tópicos

Criar o aplicativo

Para criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
2. Verifique se a região correta está selecionada: us-east-1 Leste dos EUA (Norte da Virgínia)

3. Abra o menu à direita e escolha aplicativos Apache Flink e, em seguida, Criar aplicativo de streaming. Como alternativa, escolha Criar aplicativo de streaming no contêiner Começar da página inicial.
4. Na página Criar aplicativo de streaming:
   - Escolha um método para configurar o aplicativo de processamento de fluxo: escolha Criar do zero.
   - Configuração do Apache Flink, versão do aplicativo Flink: escolha Apache Flink 1.19.
5. Configure seu aplicativo
   - Nome do aplicativo: enter**`MyApplication`**.
   - Descrição: enter**`My java test app`**.
   - Acesso aos recursos do aplicativo: escolha Criar/atualizar IAM função **`kinesis-analytics-MyApplication-us-east-1`** com as políticas necessárias.
6. Configure seu modelo para as configurações do aplicativo
   - Modelos: escolha Desenvolvimento.
7. Escolha Criar aplicativo de streaming na parte inferior da página.

> **Note**
>
> Ao criar um serviço gerenciado para o aplicativo Apache Flink usando o console, você tem a opção de criar uma IAM função e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses IAM recursos são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-east-1`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-east-1`*
>
> O Amazon Managed Service para Apache Flink era conhecido anteriormente como Kinesis Data Analytics. O nome dos recursos que são criados automaticamente é prefixado com `kinesis-analytics-` para fins de compatibilidade com versões anteriores.

Edite a IAM política

Edite a IAM política para adicionar permissões para acessar os fluxos de dados do Kinesis.

Para editar a política

1. Abra o IAM console em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **kinesis-analytics-service-MyApplication-us-east-1** que o console criou para você na seção anterior.
3. Escolha Editar e, em seguida, escolha a JSONguia.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua a conta de amostra IDs (*012345678901*) com o ID da sua conta.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/kinesis-analytics-placeholder-s3-object"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
```

```
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

5. Escolha Avançar na parte inferior da página e escolha Salvar alterações.

Configurar o aplicativo

Edite a configuração do aplicativo para definir o artefato do código do aplicativo.

Para editar a configuração

1. Na MyApplicationpágina, escolha Configurar.

2. Na seção Localização do código do aplicativo:
   - Para o bucket do Amazon S3, selecione o bucket que você criou anteriormente para o código do aplicativo. Escolha Procurar e selecione o bucket correto e, em seguida, selecione Escolher. Não clique no nome do bucket.
   - Em Caminho do objeto do Amazon S3, insira **`amazon-msf-java-stream-app-1.0.jar`**.
3. Para permissões de acesso, escolha Criar/atualizar IAM função **`kinesis-analytics-MyApplication-us-east-1`** com as políticas necessárias.
4. Na seção Propriedades do tempo de execução, adicione as propriedades a seguir.
5. Escolha Adicionar novo item e adicione cada um dos seguintes parâmetros:

ID do grupo	Chave	Valor
`InputStream0`	**`stream.name`**	**`ExampleInputStream`**
`InputStream0`	**`aws.region`**	**`us-east-1`**
`OutputStream0`	**`stream.name`**	**`ExampleOutputStream`**
`OutputStream0`	**`aws.region`**	**`us-east-1`**

6. Não modifique nenhuma das outras seções.
7. Escolha Salvar alterações.

> **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`

Execute o aplicativo

O aplicativo agora está configurado e pronto para ser executado.

Executar o aplicativo

1. No console do Amazon Managed Service para Apache Flink, escolha Meu aplicativo e escolha Executar.
2. Na próxima página, na página de configuração de restauração do aplicativo, escolha Executar com o snapshot mais recente e, em seguida, escolha Executar.

   O status nos detalhes do aplicativo muda de `Ready` para `Starting` e depois para `Running` quando o aplicativo é iniciado.

Quando o aplicativo está no `Running` status, agora você pode abrir o painel do Flink.

Para abrir o painel do

1. Escolha Abrir painel do Apache Flink. O painel é aberto em uma nova página.
2. Na lista de trabalhos em execução, escolha o único trabalho que você pode ver.

> **Note**
>
> Se você definiu as propriedades do Runtime ou editou as IAM políticas incorretamente, o status do aplicativo pode se transformar em`Running`, mas o painel do Flink mostra que o trabalho está sendo reiniciado continuamente. Esse é um cenário de falha comum se o aplicativo estiver configurado incorretamente ou não tiver permissões para acessar os recursos externos.
> Quando isso acontecer, verifique a guia Exceções no painel do Flink para ver a causa do problema.

Observe as métricas do aplicativo em execução

Na MyApplicationpágina, na seção de CloudWatch métricas da Amazon, você pode ver algumas das métricas fundamentais do aplicativo em execução.

Para ver as métricas

1. Ao lado do botão Atualizar, selecione 10 segundos na lista suspensa.
2. Quando o aplicativo está em execução e em bom estado, você pode ver a métrica de tempo de atividade aumentando continuamente.

3. A métrica de reinicializações completas deve ser zero. Se estiver aumentando, a configuração pode ter problemas. Para investigar o problema, revise a guia Exceções no painel do Flink.
4. A métrica Número de pontos de verificação com falha deve ser zero em um aplicativo saudável.

> **Note**
>
> Esse painel exibe um conjunto fixo de métricas com uma granularidade de 5 minutos. Você pode criar um painel de aplicativos personalizado com qualquer métrica no CloudWatch painel.

Observe os dados de saída nos streams do Kinesis

Verifique se você ainda está publicando dados na entrada, usando o script Python ou o Kinesis Data Generator.

Agora você pode observar a saída do aplicativo em execução no Managed Service for Apache Flink usando o Visualizador de dados no https://console.aws.amazon.com/kinesis/, da mesma forma que você já fez anteriormente.

Para ver a saída

1. Abra o console do Kinesis em https://console.aws.amazon.com /kinesis.
2. Verifique se a região é a mesma que você está usando para executar este tutorial. Por padrão, é US-East-1US East (Norte da Virgínia). Altere a região, se necessário.
3. Escolha fluxos de dados.
4. Selecione o stream que você deseja observar. Para este tutorial, use `ExampleOutputStream`.
5. Escolha a guia Visualizador de dados.
6. Selecione qualquer fragmento, mantenha Último como posição inicial e escolha Obter registros. Talvez você veja o erro “nenhum registro encontrado para esta solicitação”. Em caso afirmativo, escolha Tentar obter registros novamente. Os registros mais recentes publicados no stream são exibidos.
7. Selecione o valor na coluna Dados para inspecionar o conteúdo do registro em JSON formato.

Pare o aplicativo

Para interromper o aplicativo, acesse a página do console do aplicativo Managed Service for Apache Flink chamada. `MyApplication`

Como interromper o aplicativo

1. Na lista suspensa Ação, escolha Parar.
2. O status nos detalhes do aplicativo muda de `Running` para `Stopping` e depois para `Ready` quando o aplicativo é completamente interrompido.

> **Note**
>
> Não se esqueça também de parar de enviar dados para o stream de entrada a partir do script Python ou do Kinesis Data Generator.

Próxima etapa

Limpe AWS os recursos

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados neste tutorial de Getting Started (DataStream API).

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seus objetos e bucket do Amazon S3
- Exclua seus IAM recursos
- Exclua seus CloudWatch recursos
- Explore recursos adicionais para o Apache Flink

Exclua seu aplicativo Managed Service for Apache Flink

Siga o procedimento a seguir 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 lista suspensa Ações, escolha Excluir e confirme a exclusão.

Exclua seus streams de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em /flink. https://console.aws.amazon.com
2. Escolha fluxos de dados.
3. Selecione os dois fluxos que você criou `ExampleInputStream` e. `ExampleOutputStream`
4. Na lista suspensa Ações, escolha Excluir e confirme a exclusão.

Exclua seus objetos e bucket do Amazon S3

Use os procedimentos a seguir para excluir seus objetos e bucket do Amazon S3.

Para excluir o objeto do bucket do S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Selecione o bucket do S3 que você criou para o artefato do aplicativo.
3. Selecione o artefato do aplicativo que você carregou, chamado`amazon-msf-java-stream-app-1.0.jar`.
4. Selecione Excluir para confirmar a exclusão.

Para excluir o bucket do S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Selecione o bucket que você criou para os artefatos.
3. Selecione Excluir para confirmar a exclusão.

> **Note**
>
> O bucket do S3 deve estar vazio para excluí-lo.

Exclua seus IAM recursos

Para excluir seus IAM recursos

1. Abra o IAM console 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-east-1.
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-east-1.
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.

Explore recursos adicionais para o Apache Flink

Explore recursos adicionais

Explore recursos adicionais

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.

- 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.

- Exemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink: 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.
- Aprenda Flink: treinamento prático: treinamento introdutório oficial do Apache Flink que ajuda você a começar a escrever aplicativos escaláveis de streamingETL, análises e orientados a eventos.

Comece a usar o Amazon Managed Service para Apache Flink (tabela) API

Esta seção apresenta os conceitos fundamentais do Managed Service for Apache Flink e da implementação de um aplicativo em Java usando a Tabela e. API SQL Ele demonstra como alternar entre diferentes APIs no mesmo aplicativo e 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

Analise os componentes do aplicativo Managed Service for Apache Flink

> **Note**
>
> O Managed Service para Apache Flink é compatível com todo o Apache Flink APIs e, potencialmente, com todos os idiomas. JVM Dependendo da API sua escolha, a estrutura do aplicativo e a implementação são um pouco diferentes. Este tutorial aborda a implementação de aplicativos usando a Tabela API eSQL, e a integração com a DataStream API, implementada em Java.

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um aplicativo Java que processa a entrada e produz a saída usando o tempo de execução do Apache Flink.

Um aplicativo Apache Flink típico tem os seguintes componentes:

- Propriedades de tempo de execução: você pode usar propriedades de tempo de execução para passar parâmetros de configuração para seu aplicativo sem modificar e republicar o código.
- Fontes: o aplicativo consome dados de uma ou mais fontes. Uma fonte usa um conector para ler dados de um sistema externo, como um stream de dados do Kinesis ou um tópico da AmazonMSK. Para desenvolvimento ou teste, você também pode fazer com que as fontes gerem dados de teste aleatoriamente. Para obter mais informações, consulte Adicione fontes de dados de streaming ao Managed Service for Apache Flink. Com SQL ou TabelaAPI, as fontes são definidas como tabelas de origem.
- Transformações: o aplicativo processa dados por meio de uma ou mais transformações que podem filtrar, enriquecer ou agregar dados. Ao usar SQL ou TabelaAPI, as transformações são definidas como consultas em tabelas ou visualizações.
- Coletores: o aplicativo envia dados para sistemas externos por meio de coletores. Um coletor usa um conector para enviar dados para um sistema externo, como um stream de dados do Kinesis, um MSK tópico da Amazon, um bucket do Amazon S3 ou um banco de dados relacional. Você também pode usar um conector especial para imprimir a saída somente para fins de desenvolvimento. Ao usar SQL ou TabelaAPI, os coletores são definidos como tabelas de coletores nas quais você inserirá os resultados. Para obter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

Seu aplicativo requer algumas dependências externas, como conectores Flink que seu aplicativo usa ou, potencialmente, uma biblioteca Java. Para ser executado no Amazon Managed Service para Apache Flink, você deve empacotar o aplicativo junto com as dependências em um fat- JAR e enviá-lo para um bucket do Amazon S3. Em seguida, crie um aplicativo do Managed Service for Apache Flink. Você passa a localização do pacote de código, junto com outros parâmetros de configuração de tempo de execução. Este tutorial demonstra como usar o Apache Maven para empacotar o aplicativo e como executá-lo localmente no local IDE de sua escolha.

Preencha os pré-requisitos necessários

Antes de iniciar este tutorial, conclua duas primeiras etapas de Comece a usar o Amazon Managed Service para Apache Flink () DataStream API:

- Cumpra os pré-requisitos para concluir os exercícios
- Configure o AWS Command Line Interface (AWS CLI)

Para começar, consulte o Cria uma aplicação.

Crie e execute um serviço gerenciado para o aplicativo Apache Flink

Neste exercício, você cria um serviço gerenciado para o aplicativo Apache Flink com fluxos de dados do Kinesis como fonte e coletor.

Esta seção contém as seguintes etapas.

- Crie recursos dependentes
- Configurar seu ambiente de desenvolvimento local
- Baixe e examine o código Java de streaming do Apache Flink
- Execute seu aplicativo localmente
- Observe o aplicativo gravando dados em um bucket S3
- Interrompa a execução local do seu aplicativo
- Compile e empacote o código do seu aplicativo
- Faça o upload do JAR arquivo de código do aplicativo
- Crie e configure o serviço gerenciado para o aplicativo Apache Flink

Crie recursos dependentes

Antes de criar um Managed Service for Apache Flink para este exercício, você cria os seguintes recursos dependentes:

- Um bucket do Amazon S3 para armazenar o código do aplicativo e gravar a saída do aplicativo.

> **Note**
>
> Este tutorial pressupõe que você esteja implantando seu aplicativo na região us-east-1. Se você usa outra região, deve adaptar todas as etapas adequadamente.

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.

> **Note**
>
> Certifique-se de criar o bucket na região que você usa para este tutorial. O padrão para o tutorial é us-east-1.

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/<my-application>`.
- Um fluxo de logs chamado `kinesis-analytics-log-stream`.

Configurar seu ambiente de desenvolvimento local

Para desenvolvimento e depuração, você pode executar o aplicativo Apache Flink em sua máquina, diretamente da sua escolha. IDE Todas as dependências do Apache Flink são tratadas como dependências normais do Java usando o Maven.

> **Note**
>
> Em sua máquina de desenvolvimento, você deve ter o Java JDK 11, o Maven e o Git instalados. Recomendamos que você use um ambiente de desenvolvimento como o Eclipse, Java Neon ou IntelliJ. IDEA Para verificar se você atende a todos os pré-requisitos, consulte. Cumpra os pré-requisitos para concluir os exercícios Você não precisa instalar um cluster Apache Flink em sua máquina.

Autentique sua sessão AWS

O aplicativo usa fluxos de dados do Kinesis para publicar dados. Ao executar localmente, você deve ter uma sessão AWS autenticada válida com permissões para gravar no stream de dados do Kinesis. Use as etapas a seguir para autenticar sua sessão:

1. Se você não tiver o AWS CLI e um perfil nomeado com credencial válida configurado, consulteConfigure o AWS Command Line Interface (AWS CLI).
2. Se você IDE tiver um plug-in com o qual se integrar AWS, poderá usá-lo para passar as credenciais para o aplicativo em execução noIDE. Para obter mais informações, consulte AWS Kit de ferramentas para IDEA IntelliJ AWS e Kit de ferramentas para compilar o aplicativo ou executar o Eclipse.

Baixe e examine o código Java de streaming do Apache Flink

O código do aplicativo 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-managed-service-for-apache-flink-examples.git
```

2. Navegue até o diretório `./java/GettingStartedTable`.

Revise os componentes do aplicativo

O aplicativo é totalmente implementado na `com.amazonaws.services.msf.BasicTableJob` classe. O `main()` método define fontes, transformações e sumidouros. A execução é iniciada por uma instrução de execução no final desse método.

> **Note**
>
> Para uma experiência ideal para o desenvolvedor, o aplicativo foi projetado para ser executado sem nenhuma alteração de código no Amazon Managed Service para Apache Flink e localmente, para desenvolvimento em seu. IDE

- Para ler a configuração do tempo de execução para que ela funcione durante a execução no Amazon Managed Service para Apache Flink e no seuIDE, o aplicativo detecta automaticamente se está sendo executado de forma independente localmente no. IDE Nesse caso, o aplicativo carrega a configuração do tempo de execução de forma diferente:

1. Quando o aplicativo detectar que está sendo executado no modo autônomo no seuIDE, forme o `application_properties.json` arquivo incluído na pasta de recursos do projeto. O conteúdo do arquivo é apresentado a seguir.
2. Quando o aplicativo é executado no Amazon Managed Service para Apache Flink, o comportamento padrão carrega a configuração do aplicativo a partir das propriedades de tempo de execução que você definirá no aplicativo Amazon Managed Service para Apache Flink. Consulte Crie e configure o serviço gerenciado para o aplicativo Apache Flink.

```
private static Map<String, Properties>
 loadApplicationProperties(StreamExecutionEnvironment env) throws IOException {
    if (env instanceof LocalStreamEnvironment) {
        LOGGER.info("Loading application properties from '{}'",
 LOCAL_APPLICATION_PROPERTIES_RESOURCE);
        return KinesisAnalyticsRuntime.getApplicationProperties(
                BasicStreamingJob.class.getClassLoader()

 .getResource(LOCAL_APPLICATION_PROPERTIES_RESOURCE).getPath());
    } else {
        LOGGER.info("Loading application properties from Amazon Managed Service for
 Apache Flink");
        return KinesisAnalyticsRuntime.getApplicationProperties();
    }
}
```

- O `main()` método define o fluxo de dados do aplicativo e o executa.
  - Inicializa os ambientes de streaming padrão. Neste exemplo, mostramos como criar tanto o `StreamExecutionEnvironment` para usar com o DataStream API quanto o `StreamTableEnvironment` para usar com SQL e a TabelaAPI. Os dois objetos de ambiente são duas referências separadas ao mesmo ambiente de tempo de execução, para uso diferenteAPIs.

```
StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();
StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env,
 EnvironmentSettings.newInstance().build());
```

  - Carregue os parâmetros de configuração do aplicativo. Isso os carregará automaticamente do local correto, dependendo de onde o aplicativo está sendo executado:

```
Map<String, Properties> applicationParameters = loadApplicationProperties(env);
```

- O conector FileSystem coletor que o aplicativo usa para gravar resultados nos arquivos de saída do Amazon S3 quando o Flink conclui um ponto de verificação. Você deve ativar os pontos de verificação para gravar arquivos no destino. Quando o aplicativo está sendo executado no Amazon Managed Service para Apache Flink, a configuração do aplicativo controla o ponto de verificação e o ativa por padrão. Por outro lado, quando executados localmente, os pontos de verificação são desativados por padrão. O aplicativo detecta que ele é executado localmente e configura o ponto de verificação a cada 5.000 ms.

```
if (env instanceof LocalStreamEnvironment) {
   env.enableCheckpointing(5000);
}
```

- Esse aplicativo não recebe dados de uma fonte externa real. Ele gera dados aleatórios para serem processados por meio do DataGen conector. Esse conector está disponível para DataStream APISQL, e TabelaAPI. Para demonstrar a integração entre APIs eles, o aplicativo usa a DataStram API versão porque ela oferece mais flexibilidade. Cada registro é gerado por uma função geradora chamada `StockPriceGeneratorFunction` nesse caso, na qual você pode colocar uma lógica personalizada.

```
DataGeneratorSource<StockPrice> source = new DataGeneratorSource<>(
        new StockPriceGeneratorFunction(),
        Long.MAX_VALUE,
        RateLimiterStrategy.perSecond(recordPerSecond),
        TypeInformation.of(StockPrice.class));
```

- No DataStream API, os registros podem ter classes personalizadas. As aulas devem seguir regras específicas para que o Flink possa usá-las como registro. Para obter mais informações, consulte Tipos de dados compatíveis. Neste exemplo, a `StockPrice` classe é uma POJO.
- A fonte é então anexada ao ambiente de execução, gerando um `DataStream` de`StockPrice`. Esse aplicativo não usa semântica de tempo de evento e não gera uma marca d'água. Execute a DataGenerator fonte com um paralelismo de 1, independente do paralelismo do resto do aplicativo.

```
DataStream<StockPrice> stockPrices = env.fromSource(
        source,
        WatermarkStrategy.noWatermarks(),
```

```
        "data-generator"
    ).setParallelism(1);
```

- O que segue no fluxo de processamento de dados é definido usando a Tabela API SQL e. Para fazer isso, convertemos o DataStream of StockPrices em uma tabela. O esquema da tabela é automaticamente inferido da `StockPrice` classe.

```
Table stockPricesTable = tableEnv.fromDataStream(stockPrices);
```

- O trecho de código a seguir mostra como definir uma visualização e uma consulta usando a tabela programática: API

```
Table filteredStockPricesTable = stockPricesTable.
        select(
                $("eventTime").as("event_time"),
                $("ticker"),
                $("price"),
                dateFormat($("eventTime"), "yyyy-MM-dd").as("dt"),
                dateFormat($("eventTime"), "HH").as("hr")
        ).where($("price").isGreater(50));

tableEnv.createTemporaryView("filtered_stock_prices", filteredStockPricesTable);
```

- Uma tabela de coletor é definida para gravar os resultados em um bucket do Amazon S3 como JSON arquivos. Para ilustrar a diferença de definir uma visualização programaticamente, com a Tabela, API a tabela coletora é definida usando. SQL

```
tableEnv.executeSql("CREATE TABLE s3_sink (" +
            "eventTime TIMESTAMP(3)," +
            "ticker STRING," +
            "price DOUBLE," +
            "dt STRING," +
            "hr STRING" +
        ") PARTITIONED BY ( dt, hr ) WITH (" +
        "'connector' = 'filesystem'," +
        "'fmat' = 'json'," +
        "'path' = 's3a://"  + s3Path + "'" +
        ")");
```

- A última etapa do é inserir `executeInsert()` a visualização filtrada dos preços das ações na mesa da pia. Esse método inicia a execução do fluxo de dados que definimos até agora.

```
filteredStockPricesTable.executeInsert("s3_sink");
```

Use o arquivo pom.xml

O arquivo pom.xml define todas as dependências exigidas pelo aplicativo e configura o plug-in Maven Shade para criar o fat-jar que contém todas as dependências exigidas pelo Flink.

- Algumas dependências têm `provided` escopo. Essas dependências estão disponíveis automaticamente quando o aplicativo é executado no Amazon Managed Service para Apache Flink. Eles são necessários para a inscrição ou para o aplicativo local em seuIDE. Para obter mais informações, consulte (atualizar para a tabelaAPI)Execute seu aplicativo localmente. Certifique-se de usar a mesma versão do Flink do tempo de execução que você usará no Amazon Managed Service para Apache Flink. Para usar a Tabela API eSQL, você deve incluir o `flink-table-planner-loader` e`flink-table-runtime-dependencies`, ambos com `provided` escopo.

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-clients</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-planner-loader</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-runtime</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
```

- Você deve adicionar dependências adicionais do Apache Flink ao pom com o escopo padrão. Por exemplo, o DataGen conector, o FileSystem SQLconector e o JSONformato.

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-datagen</artifactId>
    <version>${flink.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-files</artifactId>
    <version>${flink.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-json</artifactId>
    <version>${flink.version}</version>
</dependency>
```

- Para gravar no Amazon S3 quando executado localmente, o S3 Hadoop File System também está incluído no escopo. `provided`

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-s3-fs-hadoop</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
```

- O plug-in Maven Java Compiler garante que o código seja compilado em Java 11, a JDK versão atualmente suportada pelo Apache Flink.
- O plug-in Maven Shade empacota o fat-jar, excluindo algumas bibliotecas que são fornecidas pelo tempo de execução. Também especifica dois transformadores: e. `ServicesResourceTransformer` `ManifestResourceTransformer` O último configura a classe que contém o `main` método para iniciar o aplicativo. Se você renomear a classe principal, não se esqueça de atualizar esse transformador.
-

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
```

```
    ...
        <transformer
 implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
            <mainClass>com.amazonaws.services.msf.BasicStreamingJob</mainClass>
        </transformer>
    ...
</plugin>
```

Execute seu aplicativo localmente

Você pode executar e depurar seu aplicativo Flink localmente em seu. IDE

> **Note**
>
> Antes de continuar, verifique se os fluxos de entrada e saída estão disponíveis. Consulte Crie dois streams de dados do Amazon Kinesis. Além disso, verifique se você tem permissão para ler e gravar nos dois fluxos. Consulte Autentique sua sessão AWS.
> A configuração do ambiente de desenvolvimento local requer Java 11JDK, Apache Maven e an IDE para desenvolvimento em Java. Verifique se você atende aos pré-requisitos exigidos. Consulte Cumpra os pré-requisitos para concluir os exercícios.

Importe o projeto Java para o seu IDE

Para começar a trabalhar no aplicativo em seuIDE, você deve importá-lo como um projeto Java.

O repositório que você clonou contém vários exemplos. Cada exemplo é um projeto separado. Para este tutorial, importe o conteúdo do `./jave/GettingStartedTable` subdiretório para o seuIDE.

Insira o código como um projeto Java existente usando o Maven.

> **Note**
>
> O processo exato para importar um novo projeto Java varia de acordo com o IDE que você está usando.

Modificar a configuração do aplicativo local

Ao ser executado localmente, o aplicativo usa a configuração no `application_properties.json` arquivo na pasta de recursos do projeto abaixo `./src/main/resources`. Para este aplicativo tutorial, os parâmetros de configuração são o nome do bucket e o caminho em que os dados serão gravados.

Edite a configuração e modifique o nome do bucket do Amazon S3 para corresponder ao bucket que você criou no início deste tutorial.

```
[
 {
 "PropertyGroupId": "bucket",
 "PropertyMap": {
 "name": "<bucket-name>",
 "path": "output"
 }
 }
]
```

> **Note**
>
> A propriedade de configuração `name` deve conter somente o nome do bucket, por exemplo`my-bucket-name`. Não inclua nenhum prefixo, como `s3://` ou uma barra final. Se você modificar o caminho, omita as barras à esquerda ou à direita.

Defina sua configuração de IDE execução

Você pode executar e depurar o aplicativo Flink IDE diretamente de você executando a classe principal`com.amazonaws.services.msf.BasicTableJob`, da mesma forma que executaria qualquer aplicativo Java. Antes de executar o aplicativo, você deve definir a configuração Executar. A configuração depende do IDE que você está usando. Por exemplo, consulte Configurações de execução/depuração na documentação do IntelliJ. IDEA Em particular, você deve configurar o seguinte:

1. Adicione as **`provided`** dependências ao classpath. Isso é necessário para garantir que as dependências com `provided` escopo sejam passadas para o aplicativo quando executado localmente. Sem essa configuração, o aplicativo exibe um `class not found` erro imediatamente.

2. Passe as AWS credenciais para acessar os streams do Kinesis para o aplicativo. A maneira mais rápida é usar o AWS Toolkit for IDEA IntelliJ. Usando esse IDE plug-in na configuração Executar, você pode selecionar um AWS perfil específico. AWS a autenticação acontece usando esse perfil. Você não precisa passar as AWS credenciais diretamente.
3. Verifique se ele IDE executa o aplicativo usando JDK11.

Execute o aplicativo em seu IDE

Depois de definir a configuração Executar para o`BasicTableJob`, você pode executá-la ou depurá-la como um aplicativo Java comum.

> **Note**
>
> Você não pode executar o fat-jar gerado pelo Maven diretamente na linha `java -jar ...` de comando. Esse jar não contém as dependências principais do Flink necessárias para executar o aplicativo de forma independente.

Quando o aplicativo é iniciado com êxito, ele registra algumas informações sobre o minicluster autônomo e a inicialização dos conectores. Isso é seguido por vários INFO e alguns WARN registros que o Flink normalmente emite quando o aplicativo é iniciado.

```
21:28:34,982 INFO  com.amazonaws.services.msf.BasicTableJob
               [] - Loading application properties from 'flink-application-properties-
dev.json'
21:28:35,149 INFO  com.amazonaws.services.msf.BasicTableJob
[] - s3Path is ExampleBucket/my-output-bucket
...
```

Depois que a inicialização for concluída, o aplicativo não emitirá mais nenhuma entrada de registro. Enquanto os dados estão fluindo, nenhum registro é emitido.

Para verificar se o aplicativo está processando dados corretamente, você pode inspecionar o conteúdo do bucket de saída, conforme descrito na seção a seguir.

> **Note**
>
> Não emitir registros sobre o fluxo de dados é o comportamento normal de um aplicativo Flink. A emissão de registros em cada registro pode ser conveniente para depuração, mas pode adicionar uma sobrecarga considerável durante a execução em produção.

Observe o aplicativo gravando dados em um bucket S3

Esse aplicativo de exemplo gera dados aleatórios internamente e grava esses dados no bucket S3 de destino que você configurou. A menos que você tenha modificado o caminho de configuração padrão, os dados serão gravados no `output` caminho seguido pelo particionamento de dados e horas, no formato. `./output/<yyyy-MM-dd>/<HH>`

O conector do FileSystem coletor cria novos arquivos no ponto de verificação do Flink. Ao ser executado localmente, o aplicativo executa um ponto de verificação a cada 5 segundos (5.000 milissegundos), conforme especificado no código.

```
if (env instanceof LocalStreamEnvironment) {
   env.enableCheckpointing(5000);
}
```

Para navegar no bucket do S3 e observar o arquivo gravado pelo aplicativo

1. 1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Escolha o bucket que você criou anteriormente.
3. Navegue até o `output` caminho e, em seguida, até as pastas de data e hora que correspondem à hora atual no fuso UTC horário.
4. Atualize periodicamente para observar novos arquivos aparecendo a cada 5 segundos.
5. Selecione e baixe um arquivo para observar o conteúdo.

> **Note**
>
> Por padrão, os arquivos não têm extensões. O conteúdo é formatado comoJSON. Você pode abrir os arquivos com qualquer editor de texto para inspecionar o conteúdo.

Interrompa a execução local do seu aplicativo

Pare a execução do aplicativo em seuIDE. IDEGeralmente fornece uma opção de “parar”. A localização e o método exatos dependem doIDE.

Compile e empacote o código do seu aplicativo

Nesta seção, você usa o Apache Maven para compilar o código Java e empacotá-lo em um arquivo. JAR Você pode compilar e empacotar seu código usando a ferramenta de linha de comando Maven ou seu. IDE

Para compilar e empacotar usando a linha de comando do Maven

Vá para o diretório que contém o GettingStarted projeto Jave e execute o seguinte comando:

```
$ mvn package
```

Para compilar e empacotar usando seu IDE

Execute `mvn package` a partir da sua integração com o IDE Maven.

Em ambos os casos, o JAR arquivo `target/amazon-msf-java-table-app-1.0.jar` é criado.

> **Note**
>
> Executar um projeto de compilação a partir do seu IDE pode não criar o JAR arquivo.

Faça o upload do JAR arquivo de código do aplicativo

Nesta seção, você carrega o JAR arquivo criado na seção anterior para o bucket do Amazon S3 que você criou no início deste tutorial. Se você já fez isso, conclua Criar um bucket do Amazon S3.

Para fazer upload do código do aplicativo

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Escolha o bucket que você criou anteriormente para o código do aplicativo.
3. Escolha o campo Carregar.
4. Escolha Adicionar arquivos.

5. Navegue até o JAR arquivo gerado na seção anterior:`target/amazon-msf-java-table-app-1.0.jar`.
6. Escolha Carregar sem alterar nenhuma outra configuração.

> **Warning**
>
> Certifique-se de selecionar o JAR arquivo correto em`<repo-dir>/java/GettingStarted/target/`**`amazon/msf-java-table-app-1.0.jar`**.
> O diretório de destino também contém outros JAR arquivos que você não precisa carregar.

Crie e configure o serviço gerenciado para o aplicativo Apache Flink

Você pode criar e configurar um serviço gerenciado para o aplicativo Apache Flink usando o console ou o. AWS CLI Para este tutorial, você usará o console.

> **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. Ao criar o aplicativo usando o AWS CLI, você deve criar esses recursos separadamente.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
2. Verifique se a região correta está selecionada: Leste dos EUA (Norte da Virgínia) us-east-1.
3. No menu à direita, escolha aplicativos Apache Flink e, em seguida, escolha Criar aplicativo de streaming. Como alternativa, escolha Criar aplicativo de streaming na seção Começar da página inicial.
4. Na página Criar aplicativo de streaming, faça o seguinte:
   - Em Escolha um método para configurar o aplicativo de processamento de stream, escolha Criar do zero.
   - Para configuração do Apache Flink, versão do aplicativo Flink, escolha Apache Flink 1.19.
   - Na seção Configuração do aplicativo, preencha o seguinte:

- Em Nome do aplicativo, insira **`MyApplication`**.
- Em Descrição, insira **`My Java Table API test app`**.
- Para Acesso aos recursos do aplicativo, escolha Create/update IAM role kinesis-analytics-MyApplication -us-east-1 com as políticas necessárias.

- Em Modelo para configurações do aplicativo, faça o seguinte:
  - Em Modelos, escolha Desenvolvimento.

5. Escolha Criar aplicativo de streaming.

> **Note**
>
> Ao criar um serviço gerenciado para o aplicativo Apache Flink usando o console, você tem a opção de criar uma IAM função e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses IAM recursos são nomeados usando o nome do aplicativo e a região da seguinte forma:
>
> - Política: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-east-1`*
> - Função: `kinesisanalytics-`*`MyApplication`*`-`*`us-east-1`*

Edite a IAM política

Edite a IAM política para adicionar permissões para acessar o bucket do Amazon S3.

Para editar a IAM política para adicionar permissões de bucket do S3

1. Abra o IAM console em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-east-1`** que o console criou para você na seção anterior.
3. Escolha Editar e, em seguida, escolha a JSONguia.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua o ID da conta de amostra (*`012345678901`*) com o ID da sua conta e *`<bucket-name>`* com o nome do bucket do S3 que você criou.

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

```
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/kinesis-analytics-placeholder-s3-object"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
```

```
            "Sid": "WriteOutputBucket",
            "Effect": "Allow",
            "Action": "s3:*",
            Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        }
    ]
}
```

5. Escolha Próximo e, em seguida, escolha Salvar alterações.

Configurar o aplicativo

Edite o aplicativo para definir o artefato do código do aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na seção Localização do código do aplicativo, escolha Configurar.
   - Para o bucket do Amazon S3, selecione o bucket que você criou anteriormente para o código do aplicativo. Escolha Procurar e selecione o bucket correto e, em seguida, escolha Escolher. Não clique no nome do bucket.
   - Em Caminho do objeto do Amazon S3, insira **`amazon-msf-java-table-app-1.0.jar`**.
3. Para permissões de acesso, escolha Criar/atualizar IAM função **`kinesis-analytics-MyApplication-us-east-1`**.
4. Na seção Propriedades do tempo de execução, adicione as propriedades a seguir.
5. Escolha Adicionar novo item e adicione cada um dos seguintes parâmetros:

ID do grupo	Chave	Valor
`bucket`	`name`	**`your-bucket-name`**
`bucket`	`path`	`output`

6. Não modifique nenhuma outra configuração.
7. Escolha Salvar alterações.

> **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`

Execute o aplicativo

O aplicativo agora está configurado e pronto para ser executado.

Executar o aplicativo

1. Volte para a página do console no Amazon Managed Service para Apache Flink e escolha. MyApplication
2. Escolha Executar para iniciar o aplicativo.
3. Na configuração de restauração do aplicativo, escolha Executar com o snapshot mais recente.
4. Escolha Executar.
5. O status nos detalhes do aplicativo muda de `Ready` para `Starting` e depois para `Running` após o início do aplicativo.

Quando o aplicativo está em `Running` status, você pode abrir o painel do Flink.

Para abrir o painel e visualizar o trabalho

1. Escolha Abrir painel do Apache Flink. O painel é aberto em uma nova página.
2. Na lista de trabalhos em execução, escolha o único trabalho que você pode ver.

> **Note**
>
> Se você definiu as propriedades do tempo de execução ou editou as IAM políticas incorretamente, o status do aplicativo pode mudar para`Running`, mas o painel do Flink mostra o trabalho sendo reiniciado continuamente. Esse é um cenário de falha comum

quando o aplicativo está configurado incorretamente ou não tem as permissões para acessar os recursos externos.
Quando isso acontecer, verifique a guia Exceções no painel do Flink para investigar a causa do problema.

Observe as métricas do aplicativo em execução

Na MyApplicationpágina, na seção de CloudWatch métricas da Amazon, você pode ver algumas das métricas fundamentais do aplicativo em execução.

Para ver as métricas

1. Ao lado do botão Atualizar, selecione 10 segundos na lista suspensa.
2. Quando o aplicativo está em execução e em bom estado, você pode ver a métrica de tempo de atividade aumentando continuamente.
3. A métrica de reinicializações completas deve ser zero. Se estiver aumentando, a configuração pode ter problemas. Consulte a guia Exceções no painel do Flink para investigar o problema.
4. A métrica Número de pontos de verificação com falha deve ser zero em um aplicativo saudável.

> **Note**
>
> Esse painel exibe um conjunto fixo de métricas com uma granularidade de 5 minutos. Você pode criar um painel de aplicativos personalizado com qualquer métrica no CloudWatch painel.

Observe o aplicativo gravando dados no bucket de destino

Agora você pode observar o aplicativo em execução no Amazon Managed Service para Apache Flink gravando arquivos no Amazon S3.

Para observar os arquivos, siga o mesmo processo usado para verificar os arquivos que estavam sendo gravados quando o aplicativo estava sendo executado localmente. Consulte Observe o aplicativo gravando dados em um bucket S3.

Lembre-se de que o aplicativo grava novos arquivos no ponto de verificação do Flink. Quando executados no Amazon Managed Service para Apache Flink, os pontos de verificação são

habilitados por padrão e executados a cada 60 segundos. O aplicativo cria novos arquivos aproximadamente a cada 1 minuto.

Pare o aplicativo

Para interromper o aplicativo, acesse a página do console do aplicativo Managed Service for Apache Flink chamada. `MyApplication`

Como interromper o aplicativo

1. Na lista suspensa Ação, escolha Parar.
2. O status nos detalhes do aplicativo muda de `Running` para `Stopping` e depois para `Ready` quando o aplicativo é completamente interrompido.

> **Note**
>
> Não se esqueça também de parar de enviar dados para o stream de entrada a partir do script Python ou do Kinesis Data Generator.

Próxima etapa

Limpe AWS os recursos

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS os recursos criados no tutorial de Introdução (TabelaAPI).

Este tópico contém as seguintes seções.

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus objetos e bucket do Amazon S3
- Exclua seus IAM recursos
- Exclua seus CloudWatch recursos
- Próxima etapa

Exclua 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 lista suspensa Ações, escolha Excluir e confirme a exclusão.

Exclua seus objetos e bucket do Amazon S3

Siga o procedimento a seguir para excluir objetos e bucket do S3.

Para excluir o objeto do aplicativo do bucket do S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Selecione o bucket do S3 que você criou.
3. Selecione o nome do artefato do aplicativo que você carregou`amazon-msf-java-table-app-1.0.jar`, escolha Excluir e confirme a exclusão.

Para excluir todos os arquivos de saída gravados pelo aplicativo

1. Escolha a pasta `output`.
2. Escolha Excluir.
3. Confirme que você deseja excluir permanentemente o conteúdo.

Para excluir o bucket do S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Selecione o bucket do S3 que você criou.
3. Selecione Excluir para confirmar a exclusão.

Exclua seus IAM recursos

Use o procedimento a seguir para excluir seus IAM recursos.

Para excluir seus IAM recursos

1. Abra o IAM console 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-east-1.
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-east-1.
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

Explore recursos adicionais

Explore recursos adicionais

Agora que você criou e executou um serviço gerenciado para o aplicativo Apache Flink que usa a tabelaAPI, veja Explore recursos adicionais no. Comece a usar o Amazon Managed Service para Apache Flink () DataStream API

Comece a usar o Amazon Managed Service para Apache Flink para Python

Esta seção apresenta os conceitos fundamentais de um serviço gerenciado para Apache Flink usando Python e a tabela. API 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

Analise os componentes de um serviço gerenciado para o aplicativo Apache Flink

> **Note**
>
> O Amazon Managed Service para Apache Flink oferece suporte a todos os Apache Flink. APIs Dependendo do API que você escolher, a estrutura do aplicativo é um pouco diferente. Uma abordagem popular ao desenvolver um aplicativo Apache Flink em Python é definir o fluxo do aplicativo usando código incorporado SQL em Python. Essa é a abordagem que seguimos no tutorial de introdução a seguir.

Para processar dados, seu aplicativo Managed Service for Apache Flink usa um script Python para definir o fluxo de dados que processa a entrada e produz a saída usando o tempo de execução do Apache Flink.

Um aplicativo típico 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.

- Fontes: o aplicativo consome dados de uma ou mais fontes. Uma fonte usa um conector para ler dados de um sistema externo, como um stream de dados do Kinesis ou um tópico da AmazonMSK. Você também pode usar conectores especiais para gerar dados de dentro do aplicativo. Quando você usaSQL, o aplicativo define fontes como tabelas de origem.
- Transformações: o aplicativo processa dados usando uma ou mais transformações que podem filtrar, enriquecer ou agregar dados. Quando você usaSQL, o aplicativo define transformações como SQL consultas.
- Coletores: o aplicativo envia dados para fontes externas por meio de coletores. Um coletor usa um conector para enviar dados para um sistema externo, como um stream de dados do Kinesis, um MSK tópico da Amazon, um bucket do Amazon S3 ou um banco de dados relacional. Você também pode usar um conector especial para imprimir a saída para fins de desenvolvimento. Quando você usaSQL, o aplicativo define coletores como tabelas de coletores nas quais você insere resultados. Para obter mais informações, consulte Grave dados usando coletores no Managed Service for Apache Flink.

Seu aplicativo Python também pode exigir dependências externas, como bibliotecas Python adicionais ou qualquer conector Flink que seu aplicativo use. Ao empacotar seu aplicativo, você deve incluir todas as dependências que seu aplicativo exige. Este tutorial demonstra como incluir dependências de conectores e como empacotar o aplicativo para implantação no Amazon Managed Service para Apache Flink.

Cumpra os pré-requisitos

Para concluir este tutorial, você deve ter o seguinte:

- Python 3.11 , de preferência usando um ambiente autônomo como VirtualEnv (venv), Conda ou Miniconda.
- Cliente Git - instale o cliente Git, caso ainda não tenha feito isso.
- Java Development Kit (JDK) versão 11 - instale um Java JDK 11 e defina a variável de `JAVA_HOME` ambiente para apontar para seu local de instalação. Se você não tem um JDK 11, você pode usar Amazon Correttoou qualquer padrão JDK de nossa escolha.
  - Para verificar se você instalou JDK corretamente, execute o comando a seguir. A saída será diferente se você estiver usando um que não JDK seja o Amazon Corretto 11. Verifique se a versão é 11.x.

```
$ java --version
```

```
openjdk 11.0.23 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-11.0.23.9.1 (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.23.9.1 (build 11.0.23+9-LTS, mixed mode)
```

- Apache Maven - instale o Apache Maven se você ainda não tiver feito isso. Para obter mais informações, consulte Instalando o Apache Maven.
  - Para testar sua instalação do Apache Maven, use o seguinte comando:

```
$ mvn -version
```

> **Note**
>
> Embora seu aplicativo seja escrito em Python, o Apache Flink é executado na Java Virtual Machine (). JVM Ele distribui a maioria das dependências, como o conector Kinesis, como arquivos. JAR Para gerenciar essas dependências e empacotar o aplicativo em um ZIP arquivo, use o Apache Maven. Este tutorial explica como fazer isso.

> **Warning**
>
> Recomendamos que você use o Python 3.11 para desenvolvimento local. Essa é a mesma versão do Python usada pelo Amazon Managed Service para Apache Flink com o tempo de execução do Flink 1.19.
> A instalação da biblioteca Python Flink 1.19 no Python 3.12 pode falhar.
> Se você tiver outra versão do Python instalada por padrão em sua máquina, recomendamos que você crie um ambiente independente, como o uso do VirtualEnv Python 3.11.

IDEpara o desenvolvimento local

Recomendamos que você use um ambiente de desenvolvimento como PyCharmo Visual Studio Code para desenvolver e compilar seu aplicativo.

Em seguida, conclua as duas primeiras etapas doComece a usar o Amazon Managed Service para Apache Flink () DataStream API:

- Configurar uma AWS conta e criar um usuário administrador
- Configure o AWS Command Line Interface (AWS CLI)

Para começar, consulte o Cria uma aplicação.

Crie e execute um serviço gerenciado para o aplicativo Apache Flink para Python

Nesta seção, você cria um serviço gerenciado para o aplicativo Apache Flink para Python com um stream do Kinesis como origem e coletor.

Esta seção contém as seguintes etapas.

- Crie recursos dependentes
- Configurar seu ambiente de desenvolvimento local
- Baixe e examine o código Python de streaming do Apache Flink
- Gerenciar JAR dependências
- Grave registros de amostra no fluxo de entrada
- Execute seu aplicativo localmente
- Observe os dados de entrada e saída nos streams do Kinesis
- Pare de executar seu aplicativo localmente
- Package o código do seu aplicativo
- Faça o upload do pacote do aplicativo em um bucket do Amazon S3
- Crie e configure o serviço gerenciado para o aplicativo Apache Flink
- Próxima etapa

Crie 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 do aplicativo.

> **Note**
>
> Este tutorial pressupõe que você esteja implantando seu aplicativo na região us-east-1. Se você usa outra região, deve adaptar todas as etapas adequadamente.

Crie dois streams do Kinesis

Antes de criar um aplicativo Managed Service for Apache Flink para este exercício, crie dois streams de dados do Kinesis (`ExampleInputStreameExampleOutputStream`) na mesma região que você usará para implantar seu aplicativo (us-east-1 neste exemplo). 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 stream (`ExampleInputStream`), 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
```

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-east-1
```

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, por exemplo, anexando seu nome de login.

> **Note**
>
> Certifique-se de criar o bucket do S3 na região que você usa para este tutorial (us-east-1).

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/<my-application>`.
- Um fluxo de logs chamado `kinesis-analytics-log-stream`.

Configurar seu ambiente de desenvolvimento local

Para desenvolvimento e depuração, você pode executar o aplicativo Python Flink em sua máquina. Você pode iniciar o aplicativo a partir da linha de comando com `python main.py` ou em um Python IDE de sua escolha.

> **Note**
>
> Em sua máquina de desenvolvimento, você deve ter o Python 3.10 ou 3.11, o Java 11, o Apache Maven e o Git instalados. Recomendamos que você use um IDE como PyCharmou o Visual Studio Code. Para verificar se você atende a todos os pré-requisitos, consulte Cumpra os pré-requisitos para concluir os exercícios antes de continuar.

Instale a PyFlink biblioteca

Para desenvolver seu aplicativo e executá-lo localmente, você deve instalar a biblioteca Flink Python.

1. Crie um ambiente Python autônomo VirtualEnv usando o Conda ou qualquer ferramenta Python similar.

2. Instale a PyFlink biblioteca nesse ambiente. Use a mesma versão de tempo de execução do Apache Flink que você usará no Amazon Managed Service para Apache Flink. Atualmente, o tempo de execução recomendado é 1.19.1.

```
$ pip install apache-flink==1.19.1
```

3. Certifique-se de que o ambiente esteja ativo ao executar seu aplicativo. Se você executar o aplicativo noIDE, verifique se o IDE está usando o ambiente como tempo de execução. O processo depende do IDE que você está usando.

> **Note**
>
> Você só precisa instalar a PyFlink biblioteca. Você não precisa instalar um cluster Apache Flink em sua máquina.

Autentique sua sessão AWS

O aplicativo usa fluxos de dados do Kinesis para publicar dados. Ao executar localmente, você deve ter uma sessão AWS autenticada válida com permissões para gravar no stream de dados do Kinesis. Use as etapas a seguir para autenticar sua sessão:

1. Se você não tiver o AWS CLI e um perfil nomeado com credencial válida configurado, consulteConfigure o AWS Command Line Interface (AWS CLI).
2. Verifique se o seu AWS CLI está configurado corretamente e se seus usuários têm permissões para gravar no stream de dados do Kinesis publicando o seguinte registro de teste:

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-
key TEST
```

3. Se você IDE tiver um plug-in com o qual se integrar AWS, poderá usá-lo para passar as credenciais para o aplicativo em execução noIDE. Para obter mais informações, consulte AWS Toolkit for PyCharm, AWS Toolkit for Visual Studio Code AWS e Toolkit for IntelliJ. IDEA

Baixe e examine o código Python de streaming 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. Duplique o repositório remoto usando o seguinte comando:

```
git clone https://github.com/aws-samples/amazon-managed-service-for-apache-flink-examples.git
```

2. Navegue até o diretório `./python/GettingStarted`.

Revise os componentes do aplicativo

O código do aplicativo está localizado em`main.py`. Usamos SQL incorporado em Python para definir o fluxo do aplicativo.

> **Note**
>
> Para uma experiência otimizada do desenvolvedor, o aplicativo foi projetado para ser executado sem nenhuma alteração de código no Amazon Managed Service para Apache Flink e localmente, para desenvolvimento em sua máquina. O aplicativo usa a variável de ambiente `IS_LOCAL = true` para detectar quando está sendo executado localmente. Você deve definir a variável `IS_LOCAL = true` de ambiente no seu shell ou na configuração de execução do seuIDE.

- O aplicativo configura o ambiente de execução e lê a configuração do tempo de execução. Para funcionar tanto no Amazon Managed Service para Apache Flink quanto localmente, o aplicativo verifica a `IS_LOCAL` variável.
  - O seguinte é o comportamento padrão quando o aplicativo é executado no Amazon Managed Service para Apache Flink:
    1. Carregue as dependências empacotadas com o aplicativo. Para obter mais informações, consulte (link)
    2. Carregue a configuração das propriedades do Runtime que você define no aplicativo Amazon Managed Service for Apache Flink. Para obter mais informações, consulte (link)
  - Quando o aplicativo detecta `IS_LOCAL = true` quando você executa seu aplicativo localmente:
    1. Carrega dependências externas do projeto.
    2. Carrega a configuração do `application_properties.json` arquivo incluído no projeto.

```
...
```

```
APPLICATION_PROPERTIES_FILE_PATH = "/etc/flink/application_properties.json"
...
is_local = (
    True if os.environ.get("IS_LOCAL") else False
)
...
if is_local:
    APPLICATION_PROPERTIES_FILE_PATH = "application_properties.json"
    CURRENT_DIR = os.path.dirname(os.path.realpath(__file__))
    table_env.get_config().get_configuration().set_string(
        "pipeline.jars",
        "file:///" + CURRENT_DIR + "/target/pyflink-dependencies.jar",
    )
```

- O aplicativo define uma tabela de origem com uma CREATE TABLE declaração, usando o Kinesis Connector. Essa tabela lê dados do stream de entrada do Kinesis. O aplicativo usa o nome do fluxo, a região e a posição inicial da configuração do tempo de execução.

```
table_env.execute_sql(f"""
        CREATE TABLE prices (
                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' = '{input_stream_name}',
                'aws.region' = '{input_stream_region}',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """)
```

- O aplicativo também define uma tabela de coletor usando o Kinesis Connector neste exemplo. Essa tabela envia dados para o stream de saída do Kinesis.

```
table_env.execute_sql(f"""
            CREATE TABLE output (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3)
              )
```

```
            PARTITIONED BY (ticker)
            WITH (
              'connector' = 'kinesis',
              'stream' = '{output_stream_name}',
              'aws.region' = '{output_stream_region}',
              'sink.partitioner-field-delimiter' = ';',
              'sink.batch.max-size' = '100',
              'format' = 'json',
              'json.timestamp-format.standard' = 'ISO-8601'
            )""")
```

- Por fim, o aplicativo executa uma SQL tabela coletora `INSERT INTO...` a partir da tabela de origem. Em um aplicativo mais complexo, é provável que você tenha etapas adicionais para transformar os dados antes de gravar no coletor.

```
table_result = table_env.execute_sql("""INSERT INTO output
        SELECT ticker, price, event_time FROM prices""")
```

- Você deve adicionar outra etapa no final da `main()` função para executar o aplicativo localmente:

```
if is_local:
    table_result.wait()
```

  Sem essa instrução, o aplicativo é encerrado imediatamente quando você o executa localmente. Você não deve executar essa declaração ao executar seu aplicativo no Amazon Managed Service para Apache Flink.

Gerenciar JAR dependências

Um PyFlink aplicativo geralmente requer um ou mais conectores. O aplicativo neste tutorial usa o Kinesis Connector. Como o Apache Flink é executado em JavaJVM, os conectores são distribuídos como JAR arquivos, independentemente de você implementar seu aplicativo em Python. Você deve empacotar essas dependências com o aplicativo ao implantá-lo no Amazon Managed Service para Apache Flink.

Neste exemplo, mostramos como usar o Apache Maven para buscar as dependências e empacotar o aplicativo para ser executado no Managed Service para Apache Flink.

> **Note**
>
> Existem formas alternativas de buscar e empacotar dependências. Este exemplo demonstra um método que funciona corretamente com um ou mais conectores. Ele também permite que você execute o aplicativo localmente, para desenvolvimento e no Managed Service for Apache Flink sem alterações no código.

Use o arquivo pom.xml

O Apache Maven usa o `pom.xml` arquivo para controlar dependências e pacotes de aplicativos.

Todas JAR as dependências são especificadas no `pom.xml` arquivo do `<dependencies>...</dependencies>` bloco.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    ...
    <dependencies>
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>4.3.0-1.19</version>
        </dependency>
    </dependencies>
    ...
```

Para encontrar o artefato e a versão corretos do conector a serem usados, consulteUse conectores Apache Flink com o Managed Service para Apache Flink. Certifique-se de consultar a versão do Apache Flink que você está usando. Neste exemplo, usamos o conector Kinesis. Para o Apache Flink 1.19, a versão do conector é. `4.3.0-1.19`

> **Note**
>
> Se você estiver usando o Apache Flink 1.19, não há uma versão de conector lançada especificamente para essa versão. Use os conectores lançados para 1.18.

Dependências de download e empacotamento

Use o Maven para baixar as dependências definidas no `pom.xml` arquivo e empacotá-las para o aplicativo Python Flink.

1. Navegue até o diretório que contém o projeto Python Getting Started chamado. `python/GettingStarted`
2. Execute o seguinte comando:

```
$ mvn package
```

O Maven cria um novo arquivo chamado `./target/pyflink-dependencies.jar`. Quando você está desenvolvendo localmente em sua máquina, o aplicativo Python procura esse arquivo.

> **Note**
>
> Se você esquecer de executar este comando, ao tentar executar seu aplicativo, ele falhará com o erro: Não foi possível encontrar nenhuma fábrica para o identificador “kinesis”.

Grave registros de amostra no fluxo de entrada

Nesta seção, você enviará registros de amostra ao stream para o aplicativo processar. Você tem duas opções para gerar dados de amostra, usando um script Python ou o Kinesis Data Generator.

Gere dados de amostra usando um script Python

Você pode usar um script Python para enviar registros de amostra para o stream.

> **Note**
>
> Para executar esse script Python, você deve usar o Python 3.x e ter a biblioteca for AWS SDKPython (Boto) instalada.

Para começar a enviar dados de teste para o stream de entrada do Kinesis:

1. Baixe o script `stock.py` Python do gerador de dados no repositório do gerador GitHub de dados.

2. Execute o script `stock.py`:

```
$ python stock.py
```

Mantenha o script em execução enquanto você conclui o resto do tutorial. Agora você pode executar seu aplicativo Apache Flink.

Gere dados de amostra usando o Kinesis Data Generator

Como alternativa ao script Python, você pode usar o Kinesis Data Generator, também disponível em uma versão hospedada, para enviar dados de amostra aleatórios para o stream. O Kinesis Data Generator é executado no seu navegador e você não precisa instalar nada na sua máquina.

Para configurar e executar o Kinesis Data Generator:

1. Siga as instruções na documentação do Kinesis Data Generator para configurar o acesso à ferramenta. Você executará um AWS CloudFormation modelo que configura um usuário e uma senha.
2. Acesse o Kinesis Data Generator por meio do URL gerado pelo CloudFormation modelo. Você pode encontrar o URL na guia Saída após a conclusão do CloudFormation modelo.
3. Configure o gerador de dados:
   - Região: selecione a região que você está usando para este tutorial: us-east-1
   - Stream/stream de entrega: selecione o stream de entrada que o aplicativo usará: `ExampleInputStream`
   - Registros por segundo: 100
   - Modelo de registro: copie e cole o seguinte modelo:

```
{
  "event_time" : "{{date.now("YYYY-MM-DDTkk:mm:ss.SSSSS")}},
  "ticker" : "{{random.arrayElement(
        ["AAPL", "AMZN", "MSFT", "INTC", "TBV"]
    )}}",
  "price" : {{random.number(100)}}
}
```

4. Teste o modelo: escolha Modelo de teste e verifique se o registro gerado é semelhante ao seguinte:

```
{ "event_time" : "2024-06-12T15:08:32.04800, "ticker" : "INTC", "price" : 7 }
```

5. Inicie o gerador de dados: escolha Selecionar Enviar dados.

O Kinesis Data Generator agora está enviando dados para o. `ExampleInputStream`

Execute seu aplicativo localmente

Você pode testar o aplicativo localmente, executando a partir da linha de comando com `python main.py` ou a partir do seuIDE.

Para executar seu aplicativo localmente, você deve ter a versão correta da PyFlink biblioteca instalada conforme descrito na seção anterior. Para obter mais informações, consulte (link)

> **Note**
>
> Antes de continuar, verifique se os fluxos de entrada e saída estão disponíveis. Consulte Crie dois streams de dados do Amazon Kinesis. Além disso, verifique se você tem permissão para ler e gravar nos dois fluxos. Consulte Autentique sua sessão AWS.

Importe o projeto Python para o seu IDE

Para começar a trabalhar no aplicativo em seuIDE, você deve importá-lo como um projeto Python.

O repositório que você clonou contém vários exemplos. Cada exemplo é um projeto separado. Para este tutorial, importe o conteúdo do `./python/GettingStarted` subdiretório para o seuIDE.

Importe o código como um projeto Python existente.

> **Note**
>
> O processo exato para importar um novo projeto em Python varia de acordo com o IDE que você está usando.

Verifique a configuração do aplicativo local

Ao ser executado localmente, o aplicativo usa a configuração no `application_properties.json` arquivo na pasta de recursos do projeto abaixo`./src/main/resources`. Você pode editar esse arquivo para usar diferentes nomes ou regiões de stream do Kinesis.

```
[
  {
    "PropertyGroupId": "InputStream0",
    "PropertyMap": {
      "stream.name": "ExampleInputStream",
      "flink.stream.initpos": "LATEST",
      "aws.region": "us-east-1"
    }
  },
  {
    "PropertyGroupId": "OutputStream0",
    "PropertyMap": {
      "stream.name": "ExampleOutputStream",
      "aws.region": "us-east-1"
    }
  }
]
```

Execute seu aplicativo Python localmente

Você pode executar seu aplicativo localmente, seja na linha de comando, como um script Python normal, ou no. IDE

Para executar seu aplicativo a partir da linha de comando

1. Certifique-se de que o ambiente autônomo do Python, como o Conda VirtualEnv ou onde você instalou a biblioteca Python Flink, esteja ativo no momento.
2. Certifique-se de correr pelo `mvn package` menos uma vez.
3. Defina a `IS_LOCAL = true` variável de ambiente:

   ```
   $ export IS_LOCAL=true
   ```

4. Execute o aplicativo como um script Python normal.

```
$python main.py
```

Para executar o aplicativo de dentro do IDE

1. Configure seu IDE para executar o `main.py` script com a seguinte configuração:
    1. Use o ambiente Python independente, como o Conda VirtualEnv ou onde você instalou a biblioteca. PyFlink
    2. Use as AWS credenciais para acessar os streams de dados de entrada e saída do Kinesis.
    3. Defina `IS_LOCAL = true`.
2. O processo exato para definir a configuração de execução depende de você IDE e varia.
3. Depois de configurar o seuIDE, execute o script Python e use as ferramentas fornecidas por você IDE enquanto o aplicativo estiver em execução.

Inspecione os registros do aplicativo localmente

Quando executado localmente, o aplicativo não mostra nenhum registro no console, além de algumas linhas impressas e exibidas quando o aplicativo é iniciado. PyFlink grava registros em um arquivo no diretório em que a biblioteca Python Flink está instalada. O aplicativo imprime a localização dos registros quando é iniciado. Você também pode executar o comando a seguir para encontrar os registros:

```
$ python -c "import pyflink;import
 os;print(os.path.dirname(os.path.abspath(pyflink.__file__))+'/log')"
```

1. Liste os arquivos no diretório de registro. Normalmente, você encontra um único `.log` arquivo.
2. Acompanhe o arquivo enquanto o aplicativo estiver em execução:`tail -f <log-path>/<log-file>.log`.

Observe os dados de entrada e saída nos streams do Kinesis

Você pode observar os registros enviados ao stream de entrada pelo (gerando o Python de amostra) ou pelo Kinesis Data Generator (link) usando o Visualizador de dados no console do Amazon Kinesis.

Para observar registros:

Pare de executar seu aplicativo localmente

Pare a execução do aplicativo em seuIDE. IDEGeralmente fornece uma opção de “parar”. A localização e o método exatos dependem doIDE.

Package o código do seu aplicativo

Nesta seção, você usa o Apache Maven para empacotar o código do aplicativo e todas as dependências necessárias em um arquivo.zip.

Execute o comando do pacote Maven novamente:

```
$ mvn package
```

Esse comando gera o arquivo`target/managed-flink-pyflink-getting-started-1.0.0.zip`.

Faça o upload do pacote do aplicativo em um bucket do Amazon S3

Nesta seção, você carrega o arquivo.zip criado na seção anterior para o bucket do Amazon Simple Storage Service (Amazon S3) que você criou no início deste tutorial. Se você não concluiu essa etapa, consulte (link).

Para carregar o JAR arquivo de código do aplicativo

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Escolha o bucket que você criou anteriormente para o código do aplicativo.
3. Escolha Carregar.
4. Escolha Adicionar arquivos.
5. Navegue até o arquivo.zip gerado na etapa anterior:`target/managed-flink-pyflink-getting-started-1.0.0.zip`.
6. Escolha Carregar sem alterar nenhuma outra configuração.

Crie e configure o serviço gerenciado para o aplicativo Apache Flink

Você pode criar e configurar um serviço gerenciado para o aplicativo Apache Flink usando o console ou o. AWS CLI Para este tutorial, usaremos o console.

Criar o aplicativo

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
2. Verifique se a região correta está selecionada: Leste dos EUA (Norte da Virgínia) us-east-1.
3. Abra o menu do lado direito e escolha aplicativos Apache Flink e, em seguida, Criar aplicativo de streaming. Como alternativa, escolha Criar aplicativo de streaming na seção Começar da página inicial.
4. Na página Criar aplicativos de streaming:
   - Em Escolha um método para configurar o aplicativo de processamento de fluxo, escolha Criar do zero.
   - Para configuração do Apache Flink, versão do aplicativo Flink, escolha Apache Flink 1.19.
   - Para configuração do aplicativo:
     - Em Nome do aplicativo, insira **`MyApplication`**.
     - Em Descrição, insira **`My Python test app`**.
     - Em Acesso aos recursos do aplicativo, escolha Create/update IAM role kinesis-analytics-MyApplication -us-east-1 com as políticas necessárias.
   - Para o modelo para configurações de aplicativos:
     - Em Modelos, escolha Desenvolvimento.
   - Escolha Criar aplicativo de streaming.

> **Note**
>
> Ao criar um serviço gerenciado para o aplicativo Apache Flink usando o console, você tem a opção de criar uma IAM função e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses IAM recursos 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`*

O Amazon Managed Service para Apache Flink era conhecido anteriormente como Kinesis Data Analytics. O nome dos recursos que são gerados automaticamente é prefixado com `kinesis-analytics` para fins de compatibilidade com versões anteriores.

Edite a IAM política

Edite a IAM política para adicionar permissões para acessar o bucket do Amazon S3.

Para editar a IAM política para adicionar permissões de bucket do S3

1. Abra o IAM console em https://console.aws.amazon.com/iam/.
2. Selecione Policies (Políticas). Selecione a política **`kinesis-analytics-service-MyApplication-us-east-1`** que o console criou para você na seção anterior.
3. Escolha Editar e, em seguida, escolha a JSONguia.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua a conta de amostra IDs (*`012345678901`*) com o ID da sua conta.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/kinesis-analytics-placeholder-s3-object"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
```

```
                "arn:aws:logs:us-east-1:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

5. Escolha Próximo e, em seguida, escolha Salvar alterações.

Configurar o aplicativo

Edite a configuração do aplicativo para definir o artefato do código do aplicativo.

Configurar o aplicativo

1. Na MyApplicationpágina, escolha Configurar.
2. Na seção Localização do código do aplicativo:
   - Para o bucket do Amazon S3, selecione o bucket que você criou anteriormente para o código do aplicativo. Escolha Procurar e selecione o bucket correto e, em seguida, escolha Escolher. Não selecione o nome do bucket.
   - Em Caminho do objeto do Amazon S3, insira **`managed-flink-pyflink-getting-started-1.0.0.zip`**.
3. Para permissões de acesso, escolha Criar/atualizar IAM função **`kinesis-analytics-MyApplication-us-east-1`** com as políticas necessárias.
4. Vá para as propriedades de tempo de execução e mantenha os valores padrão para todas as outras configurações.
5. Escolha Adicionar novo item e adicione cada um dos seguintes parâmetros:

ID do grupo	Chave	Valor
`InputStream0`	**`stream.name`**	**`ExampleInputStream`**
`InputStream0`	**`flink.stream.initpos`**	**`LATEST`**
`InputStream0`	**`aws.region`**	**`us-east-1`**
`OutputStream0`	**`stream.name`**	**`ExampleOutputStream`**
`OutputStream0`	**`aws.region`**	**`us-east-1`**
`kinesis.analytics.flink.run.options`	**`python`**	**`main.py`**
`kinesis.analytics.flink.run.options`	**`jarfile`**	**`lib/pyflink-dependencies.jar`**

6. Não modifique nenhuma das outras seções e escolha Salvar alterações.

> **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`

Execute o aplicativo

O aplicativo agora está configurado e pronto para ser executado.

Executar o aplicativo

1. No console do Amazon Managed Service para Apache Flink, escolha Meu aplicativo e escolha Executar.
2. Na próxima página, na página de configuração de restauração do aplicativo, escolha Executar com o snapshot mais recente e, em seguida, escolha Executar.

   O status nos detalhes do aplicativo muda de `Ready` para `Starting` e depois para `Running` quando o aplicativo é iniciado.

Quando o aplicativo está no `Running` status, agora você pode abrir o painel do Flink.

Para abrir o painel do

1. Escolha Abrir painel do Apache Flink. O painel é aberto em uma nova página.
2. Na lista de trabalhos em execução, escolha o único trabalho que você pode ver.

> **Note**
>
> Se você definiu as propriedades do Runtime ou editou as IAM políticas incorretamente, o status do aplicativo pode se transformar em`Running`, mas o painel do Flink mostra que o trabalho está sendo reiniciado continuamente. Esse é um cenário de falha comum se

o aplicativo estiver configurado incorretamente ou não tiver permissões para acessar os recursos externos.
Quando isso acontecer, verifique a guia Exceções no painel do Flink para ver a causa do problema.

Observe as métricas do aplicativo em execução

Na MyApplicationpágina, na seção de CloudWatch métricas da Amazon, você pode ver algumas das métricas fundamentais do aplicativo em execução.

Para ver as métricas

1. Ao lado do botão Atualizar, selecione 10 segundos na lista suspensa.
2. Quando o aplicativo está em execução e em bom estado, você pode ver a métrica de tempo de atividade aumentando continuamente.
3. A métrica de reinicializações completas deve ser zero. Se estiver aumentando, a configuração pode ter problemas. Para investigar o problema, revise a guia Exceções no painel do Flink.
4. A métrica Número de pontos de verificação com falha deve ser zero em um aplicativo saudável.

> **Note**
>
> Esse painel exibe um conjunto fixo de métricas com uma granularidade de 5 minutos. Você pode criar um painel de aplicativos personalizado com qualquer métrica no CloudWatch painel.

Observe os dados de saída nos streams do Kinesis

Verifique se você ainda está publicando dados na entrada, usando o script Python ou o Kinesis Data Generator.

Agora você pode observar a saída do aplicativo em execução no Managed Service for Apache Flink usando o Visualizador de Dados no https://console.aws.amazon.com/kinesis/, da mesma forma que você já fez anteriormente.

Para ver a saída

1. Abra o console do Kinesis em https://console.aws.amazon.com /kinesis.

2. Verifique se a região é a mesma que você está usando para executar este tutorial. Por padrão, é US-East-1US East (Norte da Virgínia). Altere a região, se necessário.
3. Escolha fluxos de dados.
4. Selecione o stream que você deseja observar. Para este tutorial, use `ExampleOutputStream`.
5. Escolha a guia Visualizador de dados.
6. Selecione qualquer fragmento, mantenha Último como posição inicial e escolha Obter registros. Talvez você veja o erro “nenhum registro encontrado para esta solicitação”. Em caso afirmativo, escolha Tentar obter registros novamente. Os registros mais recentes publicados no stream são exibidos.
7. Selecione o valor na coluna Dados para inspecionar o conteúdo do registro em JSON formato.

Pare o aplicativo

Para interromper o aplicativo, acesse a página do console do aplicativo Managed Service for Apache Flink chamada. `MyApplication`

Como interromper o aplicativo

1. Na lista suspensa Ação, escolha Parar.
2. O status nos detalhes do aplicativo muda de `Running` para `Stopping` e depois para `Ready` quando o aplicativo é completamente interrompido.

> **Note**
>
> Não se esqueça também de parar de enviar dados para o stream de entrada a partir do script Python ou do Kinesis Data Generator.

Próxima etapa

Limpe AWS os recursos

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Getting Started (Python).

Este tópico contém as seguintes seções.

Exclua 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 lista suspensa Ações, escolha Excluir e confirme a exclusão.

Exclua seus streams de dados do Kinesis

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
2. Escolha fluxos de dados.
3. Selecione os dois fluxos que você criou `ExampleInputStream` e. `ExampleOutputStream`
4. Na lista suspensa Ações, escolha Excluir e confirme a exclusão.

Exclua seus objetos e bucket do Amazon S3

Siga o procedimento a seguir para excluir objetos e bucket do S3.

Para excluir o objeto do bucket do S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Selecione o bucket do S3 que você criou para o artefato do aplicativo.
3. Selecione o artefato do aplicativo que você carregou, chamado`amazon-msf-java-stream-app-1.0.jar`.

4. Selecione Excluir para confirmar a exclusão.

Para excluir o bucket do S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Selecione o bucket que você criou para os artefatos.
3. Selecione Excluir para confirmar a exclusão.

> **Note**
>
> O bucket do S3 deve estar vazio para excluí-lo.

Exclua seus IAM recursos

Use o procedimento a seguir para excluir seus IAM recursos.

Para excluir seus IAM recursos

1. Abra o IAM console 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-east-1.
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-east-1.
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.

Comece (Scala)

> **Note**
>
> A partir da versão 1.15, o Flink é gratuito para Scala. Agora, os aplicativos podem usar o Java API de qualquer versão do Scala. O Flink ainda usa o Scala em alguns componentes principais internamente, mas não expõe o Scala no carregador de classes do código do usuário. Por isso, você deve adicionar dependências do Scala em seus JAR -archives. 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 Managed Service for Apache Flink para Scala com um stream do Kinesis como origem e coletor.

Este tópico contém as seguintes seções:

Crie 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:

- Criando e atualizando 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 stream (`ExampleInputStream`), use o seguinte comando AWS CLI create-stream do Amazon Kinesis.

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

  - 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
    ```

- 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`

Grave 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 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/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 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 carrega o código do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie recursos dependentes.

Compilar o código do aplicativo

Nesta seção, você usa a ferramenta de SBTcompilação para criar o código Scala para o aplicativo. Para instalarSBT, 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 código do aplicativo, você o compila e o empacota em um JAR arquivo. Você pode compilar e empacotar seu código comSBT:

```
sbt assembly
```

2. Se o aplicativo for 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 Create bucket (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 Selecionar arquivos, selecione 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, 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 (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 /flink https://console.aws.amazon.com

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.
   - Mantenha a versão como Apache Flink versão 1.19.1.
4. Para permissões de acesso, escolha Criar/atualizar IAM função **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um serviço gerenciado para o aplicativo Apache Flink usando o console, você tem a opção de criar uma IAM função e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses IAM recursos 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 **`getting-started-scala-1.0.jar.`**.
3. Em Acesso aos recursos do aplicativo, para permissões de acesso, escolha Criar/atualizar IAM função **`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. Selecione 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`

Edite a IAM política

Edite a IAM política para adicionar permissões para acessar o bucket do Amazon S3.

Para editar a IAM política para adicionar permissões de bucket do S3

1. Abra o IAM console 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). Escolha a JSONguia.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua a conta de amostra IDs (*012345678901*) com o 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"
        }
    ]
}
```

Execute 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.

Pare o aplicativo

Para interromper o aplicativo, na MyApplicationpágina, escolha Parar. Confirme a ação.

Crie e execute o aplicativo (CLI)

Nesta seção, você usa o AWS Command Line Interface para criar e executar o aplicativo Managed Service for Apache Flink. Use o AWS CLI comando kinesisanalyticsv2 para criar e interagir com o 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 IAM recursos, seu aplicativo não poderá acessar seus fluxos de dados e registros.

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 uma IAM função (que você cria 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 Amazon Resource 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 o Tutorial: Criar e anexar sua primeira política gerenciada pelo cliente no Guia IAM do usuário.

Crie uma IAM política

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

O Managed Service for Apache Flink não pode acessar seu fluxo sem permissões. Você concede essas permissões por meio de uma IAM função. Cada IAM função 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 IAM console 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 AWS Serviço
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.

   Agora você criou uma nova IAM função chamada`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 ARN a nova função.

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

Criar o aplicativo

Salve o JSON código a seguir em um arquivo chamado`create_request.json`. Substitua ARN a função de ARN exemplo pela função que você criou anteriormente. Substitua o ARN sufixo do bucket (nome de usuário) pelo sufixo que você escolheu na seção anterior. Substitua o ID da conta de exemplo (012345678901) na função de execução do serviço pelo ID da conta.

```
{
    "ApplicationName": "getting_started",
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_19",
    "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 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.

Inicie o aplicativo

Nesta seção, você usa a StartApplicationação para iniciar o aplicativo.

Para iniciar o aplicativo

1. Salve o JSON código a seguir 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 verificar as métricas do Managed Service for Apache Flink no CloudWatch console da Amazon para verificar se o aplicativo está funcionando.

Pare o aplicativo

Nesta seção, você usa a StopApplicationação para interromper o aplicativo.

Como interromper o aplicativo

1. Salve o JSON código a seguir 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.

Atualizar propriedades do ambiente

Nesta seção, você usa a UpdateApplicationação 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 JSON código a seguir 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 precisar atualizar o código do aplicativo com uma nova versão do pacote de código, use a UpdateApplicationCLIação.

> **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 de código anterior do bucket do Amazon S3, faça o upload da nova versão e ligue`UpdateApplication`, especificando o mesmo nome de bucket e objeto 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 Crie 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"
                }
            }
        }
    }
}
```

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Tumbling Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus IAM recursos
- Exclua seus CloudWatch recursos

Exclua seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
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 streams 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, em seguida, confirme a exclusão.

Exclua seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Escolha o ka-app-code -***<username>*** balde.
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Exclua seus IAM recursos

1. Abra o IAM console 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 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.

Use o Apache Beam com o Managed Service para aplicativos Apache Flink

> **Note**
>
> O Apache Beam não é suportado na versão 1.19 do Apache Flink. Em 27 de junho de 2024, não havia nenhum Apache Flink Runner compatível com o Flink 1.18. Para obter mais informações, consulte Compatibilidade de versão do Flink na documentação do 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. Use CloudFormation com o serviço gerenciado para Apache Flink

Este tópico contém as seguintes seções:

- Limitações do Apache Flink runner com serviço gerenciado para Apache Flink
- Recursos do Apache Beam com serviço gerenciado para Apache Flink
- Crie um aplicativo usando o Apache Beam

Limitações do Apache Flink runner com serviço gerenciado para 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.

Recursos do Apache Beam com serviço gerenciado para Apache Flink

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.

Crie 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. Use o Apache Beam com o Managed Service para aplicativos Apache Flink

> **Note**
>
> Para configurar os pré-requisitos necessários para este exercício, primeiro conclua o exercício Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.

Este tópico contém as seguintes seções:

- Crie recursos dependentes
- Grave 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 de streaming do Apache Flink
- Crie e execute o serviço gerenciado para o aplicativo Apache Flink
- Limpe AWS os recursos
- Próximas etapas

Crie 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>`**.

Grave 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 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/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 ParDopara 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 Preencha os pré-requisitos necessários no tutorial Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink.
2. Compile o aplicativo com o seguinte comando:

```
mvn package -Dflink.version=1.15.2 -Dflink.version.minor=1.8
```

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

A compilação do aplicativo cria o JAR arquivo do aplicativo (`target/basic-beam-app-1.0.jar`).

Faça o upload do código Java de streaming do Apache Flink

Nesta seção, você faz o upload do seu aplicativo no bucket do Amazon S3 que você criou na seção Crie recursos dependentes.

1. No console do Amazon S3, escolha o - ka-app-code***<username>***bucket e escolha Upload.
2. Na etapa Selecionar arquivos, selecione 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 serviço gerenciado para o aplicativo 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 /flink https://console.aws.amazon.com
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**
>
> Atualmente, o Apache Beam não é compatível com a versão 1.19 ou posterior do Apache Flink.

- Selecione Apache Flink versão 1.15 no menu suspenso de versões.

4. Para permissões de acesso, escolha Criar/atualizar IAM função **`kinesis-analytics-MyApplication-us-west-2`**.
5. Selecione Create application (Criar aplicativo).

> **Note**
>
> Ao criar um serviço gerenciado para o aplicativo Apache Flink usando o console, você tem a opção de criar uma IAM função e uma política para seu aplicativo. O aplicativo usa essa função e política para acessar os recursos dependentes. Esses IAM recursos 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`*

Edite a IAM política

Edite a IAM política para adicionar permissões para acessar os fluxos de dados do Kinesis.

1. Abra o IAM console 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). Escolha a JSONguia.
4. Adicione a seção destacada do exemplo de política a seguir à política. Substitua a conta de amostra IDs (*`012345678901`*) com o ID da sua 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 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 **basic-beam-app-1.0.jar**.
3. Em Acesso aos recursos do aplicativo, para permissões de acesso, escolha Criar/atualizar IAM função **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 Monitoramento, confirme se Nível de monitoramento de métricas está definido como Aplicativo.
6. Para CloudWatch registrar, marque a caixa de seleção Ativar.
7. Selecione Atualizar.

> **Note**
>
> Quando você opta por ativar o CloudWatch registro, 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.

Execute 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.

Limpe AWS os recursos

Esta seção inclui procedimentos para limpar AWS recursos criados no tutorial Tumbling Window.

Este tópico contém as seguintes seções:

- Exclua seu aplicativo Managed Service for Apache Flink
- Exclua seus streams de dados do Kinesis
- Exclua seu objeto e bucket do Amazon S3
- Exclua seus IAM recursos
- Exclua seus CloudWatch recursos

Exclua seu aplicativo Managed Service for Apache Flink

1. Abra o console do Managed Service for Apache Flink em /flink https://console.aws.amazon.com
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 streams 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.

Exclua seu objeto e bucket do Amazon S3

1. Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/
2. Escolha o ka-app-code -***<username>*** balde.
3. Selecione Excluir e, em seguida, insira o nome do bucket para confirmar a exclusão.

Exclua seus IAM recursos

1. Abra o IAM console 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 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.

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

- Implante, opere e escale aplicativos com o Amazon Managed Service para Apache Flink
- Desenvolva aplicativos Apache Flink localmente antes de implantá-los no Managed Service for Apache Flink
- Use a detecção de eventos com o Managed Service para Apache Flink Studio
- Use a solução AWS de dados de streaming para o Amazon Kinesis
- Pratique usando um laboratório Clickstream com o Apache Flink e o Apache Kafka
- Configure o escalonamento personalizado usando o Application Auto Scaling
- Veja um exemplo de CloudWatch painel da Amazon
- Use modelos para a solução AWS de streaming de dados para a Amazon MSK
- Explore mais soluções de serviços gerenciados para Apache Flink em GitHub

Implante, opere e escale aplicativos com o Amazon Managed Service para Apache Flink

Este workshop aborda o desenvolvimento de um aplicativo Apache Flink em Java, como executar e depurar em seu aplicativo e como empacotarIDE, implantar e executar no Amazon Managed Service para Apache Flink. Você também aprenderá a escalar, monitorar e solucionar problemas em seu aplicativo.

Workshop do Amazon Managed Service para Apache Flink.

Desenvolva aplicativos Apache Flink localmente antes de implantá-los no Managed Service for Apache Flink

Este workshop demonstra os princípios básicos para começar a desenvolver aplicativos Apache Flink localmente com o objetivo de longo prazo de implantar o Managed Service for Apache Flink for Apache Flink for Apache Flink.

Guia para iniciantes sobre desenvolvimento local com o Apache Flink

Use a detecção de eventos com o Managed Service para 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

Detecção de eventos com serviço gerenciado para Apache Flink para Apache Flink

Use a solução AWS de dados de streaming para o Amazon Kinesis

A solução AWS de dados de streaming para Amazon Kinesis configura automaticamente os AWS serviços necessários para capturar, armazenar, processar e entregar dados de streaming. A solução oferece várias opções para resolver casos de uso de dados de transmissão. A opção Managed Service for Apache Flink fornece um ETL exemplo 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 AWS CloudFormation pacote para implantar o exemplo completo.
- Um CloudWatch painel para exibir as métricas do aplicativo.
- CloudWatch alarmes sobre as métricas de aplicação mais relevantes.
- Todas as IAM funções e políticas necessárias.

Solução de streaming de dados para Amazon Kinesis

Pratique usando um laboratório Clickstream com o Apache Flink e o 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.

Laboratório Clickstream

Configure o escalonamento personalizado usando o Application Auto Scaling

Dois exemplos que mostram como escalar automaticamente seu serviço gerenciado para aplicativos Apache Flink usando o Application Auto Scaling. Isso permite que você configure políticas e atributos de escalabilidade personalizados.

- 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.

Veja um exemplo de 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.

Serviço gerenciado para painel de métricas do Apache Flink

Use modelos para a solução AWS de streaming de dados para a Amazon MSK

A solução AWS de dados de streaming para a Amazon MSK fornece AWS CloudFormation modelos em que os dados fluem por produtores, armazenamento de streaming, consumidores e destinos.

AWS Solução de streaming de dados para Amazon MSK

Explore mais soluções de serviços gerenciados para 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 para Apache Flink
- Análise de sentimentos em tempo real com base no feedback do cliente

Use utilitários práticos para o Managed Service for Apache Flink

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 iniciem regularmente pontos de salvamento/instantâneos 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

Exemplos para criar e trabalhar com o Managed Service para aplicativos Apache Flink

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 funciona
- Tutorial: Comece a usar o serviço DataStream API gerenciado para Apache Flink

> **Note**
>
> Esses exemplos pressupõem que você esteja usando a região Leste dos EUA (Norte da Virgínia) (us-east-1). Se você estiver usando uma região diferente, atualize o código, os comandos e as IAM funções do aplicativo adequadamente.

Tópicos

- Exemplos de Java para Managed Service para Apache Flink
- Exemplos em Python de serviço gerenciado para Apache Flink
- Exemplos de Scala de serviço gerenciado para Apache Flink

Exemplos de Java para Managed Service para Apache Flink

Os exemplos a seguir demonstram como criar aplicativos escritos em Java.

> **Note**
>
> A maioria dos exemplos foi projetada para ser executada localmente, na sua máquina IDE de desenvolvimento e na sua preferida, e no Amazon Managed Service para Apache Flink. Eles demonstram os mecanismos que você pode usar para transmitir os parâmetros do aplicativo

e como definir a dependência corretamente para executar o aplicativo nos dois ambientes sem alterações.

Comece com o DataStream API

Este exemplo mostra um aplicativo simples, lendo de um stream de dados do Kinesis e gravando em outro stream de dados do Kinesis, usando o. `DataStream` API O exemplo demonstra como configurar o arquivo com as dependências corretas, criar o uber- e depois analisar os parâmetros de configuraçãoJAR, para que você possa executar o aplicativo localmente, no seu IDE e no Amazon Managed Service para Apache Flink.

Exemplo de código: GettingStarted

Comece com a tabela API e SQL

Este exemplo mostra um aplicativo simples usando `Table` API SQL e. Ele demonstra como integrar o `DataStream` API com o `Table` API ou SQL no mesmo aplicativo Java. Também demonstra como usar o `DataGen` conector para gerar dados de teste aleatórios de dentro do próprio aplicativo Flink, sem a necessidade de um gerador de dados externo.

Exemplo completo: GettingStartedTable

Use S3Sink () DataStream API

Este exemplo demonstra como usar o `DataStream` API's `FileSink` para gravar JSON arquivos em um bucket do S3.

Exemplo de código: S3Sink

Use uma fonte do Kinesis, padrão ou EFO consumidora, e sink () DataStream API

Este exemplo demonstra como configurar uma fonte que consome de um stream de dados do Kinesis, usando o consumidor padrão EFO ou, e como configurar um coletor no stream de dados do Kinesis.

Exemplo de código: KinesisConnectors

Use um coletor Amazon Data Firehose () DataStream API

Este exemplo mostra como enviar dados para o Amazon Data Firehose (anteriormente conhecido como Kinesis Data Firehose).

Exemplo de código: KinesisFirehoseSink

Use agregações de janelas () DataStream API

Este exemplo demonstra quatro tipos de agregação de janelas no. `DataStream` API

1. Janela deslizante com base no tempo de processamento
2. Janela deslizante com base na hora do evento
3. Janela de queda com base no tempo de processamento
4. Janela de queda com base na hora do evento

Exemplo de código: Janelamento

Usar métricas personalizadas

Este exemplo mostra como adicionar métricas personalizadas ao seu aplicativo Flink e enviá-las para CloudWatch métricas.

Exemplos de código: CustomMetrics

Exemplos em Python de serviço gerenciado para Apache Flink

Os exemplos a seguir demonstram como criar aplicativos escritos em Python.

> **Note**
>
> A maioria dos exemplos foi projetada para ser executada localmente, na sua máquina IDE de desenvolvimento e na sua preferida, e no Amazon Managed Service para Apache Flink. Eles demonstram o mecanismo simples que você pode usar para passar os parâmetros do aplicativo e como definir a dependência corretamente para executar o aplicativo nos dois ambientes sem alterações.

Dependências do projeto

A maioria dos PyFlink exemplos exige uma ou mais dependências na forma de JAR arquivos, por exemplo, para conectores Flink. Essas dependências devem então ser empacotadas com o aplicativo quando implantadas no Amazon Managed Service para Apache Flink.

Os exemplos a seguir já incluem as ferramentas que permitem executar o aplicativo localmente para desenvolvimento e teste, além de empacotar as dependências necessárias corretamente. Essas ferramentas requerem o uso de Java JDK11 e Apache Maven. Consulte o que README está contido em cada exemplo para obter instruções específicas.

Exemplos

Comece com PyFlink

Este exemplo demonstra a estrutura básica de um PyFlink aplicativo usando código SQL incorporado em Pyton. Esse projeto também fornece um esqueleto para qualquer PyFlink aplicativo que inclua JAR dependências, como conectores. A README seção fornece orientações detalhadas sobre como executar seu aplicativo Python localmente para desenvolvimento. O exemplo também mostra como incluir uma única JAR dependência, o conector SQL Kinesis neste exemplo, em PyFlink seu aplicativo.

Exemplo de código: GettingStarted

Use agregações de janelas () DataStream API

Este exemplo demonstra quatro tipos de agregação de janelas SQL incorporadas em um aplicativo Python.

1. Janela deslizante com base no tempo de processamento
2. Janela deslizante com base na hora do evento
3. Janela de queda com base no tempo de processamento
4. Janela de queda com base na hora do evento

Exemplo de código: Janelamento

Use um coletor S3

Este exemplo mostra como gravar sua saída no Amazon S3 como JSON arquivos, usando SQL incorporado em um aplicativo Python. Você deve ativar o ponto de verificação para que o coletor do S3 grave e rotacione arquivos no Amazon S3.

Exemplo de código: S3Sink

Use uma função definida pelo usuário (UDF)

Este exemplo demonstra como definir uma função definida pelo usuário, implementá-la em Python e usá-la em SQL código executado em um aplicativo Python.

Exemplo de código: UDF

Use um coletor Amazon Data Firehose

Este exemplo demonstra como enviar dados para o Amazon Data SQL Firehose usando.

Exemplo de código: FirehoseSink

Exemplos de Scala de serviço gerenciado para Apache Flink

Os exemplos a seguir demonstram como criar aplicativos usando o Scala com o Apache Flink.

Configurar um aplicativo de várias etapas

Este exemplo mostra como configurar um aplicativo Flink no Scala. Ele demonstra como configurar o SBT projeto para incluir dependências e criar o uber-. JAR

Exemplo de código: GettingStarted

Segurança no Amazon Managed Service for Apache Flink

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficiará de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar 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 AWS Serviços no escopo do programa de compatibilidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, como a confidencialidade de seus dados, os requisitos da sua organização, 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 para 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 Firehose e Amazon S3.

Criptografia de dados no Managed Service para 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 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 criptografia do lado do servidor com KMS —Managed Keys (-). 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 em execução 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 entre o Managed Service for Apache Flink e 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

AWS Identity and Access Management (IAM) é uma ferramenta Serviço da AWS que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. IAMos administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar os recursos do Managed Service for Apache Flink. IAMé um Serviço da AWS que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz 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 ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um atributo no Managed Service for Apache Flink, consulte Solução de problemas 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. Em seguida, você deve enviar solicitações ao IAM administrador para alterar as permissões dos usuários do serviço. Revise as informações nesta página para entender os conceitos básicos doIAM. Para saber mais sobre como sua empresa pode usar o IAM Managed Service para Apache Flink, consulte. Como o Amazon Managed Service para Apache Flink funciona com IAM

IAMadministrador — Se você for IAM administrador, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao Managed Service para Apache Flink. Para ver exemplos de políticas baseadas em identidade do Serviço Gerenciado para Apache Flink que você pode usar em, consulte. IAM 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 AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como IAM usuário ou assumindo uma IAM função. Usuário raiz da conta da AWS

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Os usuários (do IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você entra como uma identidade federada, seu administrador configurou previamente a federação de identidades usando IAM funções. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para você mesmo assinar solicitações, consulte Assinar AWS API solicitações no Guia IAM do usuário.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia AWS IAM Identity Center do usuário e Uso da autenticação multifator (MFA) AWS no Guia do IAMusuário.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar 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 ver a lista completa de tarefas que exigem que você faça login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do IAM usuário.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade. Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter informações sobre o IAM Identity Center, consulte O que é o IAM Identity Center? no Guia do AWS IAM Identity Center usuário.

Grupos e usuários do IAM

Um IAMusuário é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos confiar em credenciais temporárias em vez de criar IAM usuários que tenham credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com IAM os usuários, recomendamos que você alterne as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exigem credenciais de longo prazo no Guia do IAMusuário.

Um IAMgrupo é uma identidade que especifica uma coleção de IAM usuários. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar IAM recursos.

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 IAM usuário (em vez de uma função) no Guia do IAM usuário.

IAMfunções

Uma IAMfunção é uma identidade dentro da sua Conta da AWS que tem permissões específicas. É semelhante a um IAM usuário, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma IAM função no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma AWS API operação AWS CLI or ou usando uma personalizadaURL. Para obter mais informações sobre métodos de uso de funções, consulte Usando IAM funções no Guia IAM do usuário.

IAMfunções com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode 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 informações sobre funções para federação, consulte Criação de uma função para um provedor de identidade terceirizado no Guia IAM do usuário. Se você usa o IAM Identity Center, configura um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a uma função em. IAM

Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Manual do Usuário do AWS IAM Identity Center .

- Permissões temporárias IAM de IAM usuário — Um usuário ou função pode assumir uma IAM função para assumir temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas — Você pode usar uma IAM função para permitir que alguém (um diretor confiável) em uma conta diferente acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na 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 um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um IAM usuário ou uma função para realizar ações em AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um Serviço da AWS, combinadas com a solicitação Serviço da AWS para fazer solicitações aos serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.
  - Função de serviço — Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a uma Serviço da AWS no Guia do IAM usuário.
  - Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. Serviço da AWS O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.
- Aplicativos em execução na Amazon EC2 — Você pode usar uma IAM função para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e

fazendo AWS CLI AWS API solicitações. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Como usar uma IAM função para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia IAM do usuário.

Para saber se usar IAM funções ou IAM usuários, consulte Quando criar uma IAM função (em vez de um usuário) no Guia do IAM usuário.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um 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 é armazenada AWS como JSON documentos. Para obter mais informações sobre a estrutura e o conteúdo dos documentos de JSON política, consulte Visão geral das JSON políticas no Guia IAM do usuário.

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao 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 permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

IAMas políticas definem permissões para uma ação, independentemente do método usado para realizar 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 função do AWS Management Console AWS CLI, do ou do AWS API.

Políticas baseadas em identidade

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições.

Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolha entre políticas gerenciadas e políticas em linha no Guia do IAMusuário.

Políticas baseadas no recurso

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem 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 um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas de uma política baseada IAM em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Amazon S3, AWS WAF, e Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber maisACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais 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 recurso avançado no qual você define as permissões máximas que uma política baseada em identidade pode conceder a uma IAM entidade (IAMusuário ou função). 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 IAM entidades no Guia IAM do usuário.
- Políticas de controle de serviço (SCPs) — SCPs são JSON políticas que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. Os SCP limites de permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations eSCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.
- 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 atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia IAM do usuário.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de política estão envolvidos, consulte Lógica de avaliação de políticas no Guia IAM do usuário.

Como o Amazon Managed Service para Apache Flink funciona com IAM

Antes de usar IAM para gerenciar o acesso ao Managed Service for Apache Flink, saiba quais IAM recursos estão disponíveis para uso com o Managed Service for Apache Flink.

IAMrecursos que você pode usar com o Amazon Managed Service para Apache Flink

IAMrecurso	Suporte do Amazon Managed Service for Apache Flink
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de políticas	Não
ACLs	Não
ABAC(tags nas políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Não
Funções vinculadas ao serviço	Não

Para ter uma visão geral de como o Managed Service for Apache Flink e outros AWS serviços funcionam com a maioria dos IAM recursos, consulte AWS os serviços que funcionam com IAM no Guia do IAM Usuário.

Políticas baseadas em identidade para o Managed Service for Apache Flink

Compatível com políticas baseadas em identidade: Sim

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

Com políticas IAM baseadas em identidade, você pode especificar ações e recursos 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 ao qual ela está anexada. Para saber mais sobre todos os elementos que você pode usar em uma JSON política, consulte a referência IAM JSON de elementos de política no Guia IAM do usuário.

Exemplos de 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 recursos dentro do Managed Service for Apache Flink

Compatível com políticas baseadas em recursos: sim

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem 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 um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para habilitar o acesso entre contas, você pode especificar uma conta ou IAM entidades inteiras em outra conta como principal em uma política baseada em recursos. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um IAM administrador na conta confiável também deve conceder permissão à entidade principal (usuário ou função) 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 política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Ações de políticas para o Managed Service for Apache Flink

Compatível com ações de políticas: Sim

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O `Action` elemento de uma JSON política descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da AWS API operação associada. Há algumas exceções, como ações somente de permissão que não têm uma operação correspondente. API 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

Compatível com recursos de políticas: Sim

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` JSON de política especifica o objeto ou objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu Amazon Resource Name (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um 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 seusARNs, consulte Resources Defined by Amazon Managed Service for Apache Flink na Referência de Autorização de Serviço. Para saber com quais ações você pode especificar cada recurso, consulte Ações definidas pelo Amazon Managed Service para Apache Flink. ARN

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

Compatível com chaves de condição de política específicas de serviço: Sim

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes 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, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, você pode conceder permissão a um IAM usuário para acessar um recurso somente se ele estiver marcado com o nome de IAM usuário. Para obter mais informações, consulte elementos de IAM política: variáveis e tags no Guia IAM do usuário.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia IAM do usuário.

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 para Apache Flink

SuportesACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Controle de acesso baseado em atributos (ABAC) com serviço gerenciado para Apache Flink

Suportes ABAC (tags nas políticas): Sim

O controle de acesso baseado em atributos (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode

anexar tags a IAM entidades (usuários ou funções) e a muitos AWS recursos. Marcar entidades e recursos é a primeira etapa doABAC. Em seguida, você cria ABAC políticas para permitir operações quando a tag do diretor corresponde à tag do recurso que ele está tentando acessar.

ABACé útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna complicado.

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 chaves de condição `aws:TagKeys`.

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 sobreABAC, consulte O que éABAC? no Guia do IAM usuário. Para ver um tutorial com etapas de configuraçãoABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do IAMusuário.

Usando credenciais temporárias com o Managed Service for Apache Flink

Compatível com 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 esse trabalho IAM no Guia do IAM usuário.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (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 a troca de funções, consulte Alternando para uma função (console) no Guia IAM do usuário.

Você pode criar manualmente credenciais temporárias usando o AWS CLI ou AWS API. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS 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 emIAM.

Permissões de entidade principal entre serviços para o Managed Service for Apache Flink

Suporta sessões de acesso direto (FAS): Sim

Quando você usa um IAM usuário ou uma função para realizar ações em AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um Serviço da AWS, combinadas com a solicitação Serviço da AWS para fazer solicitações aos serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.

Perfis de serviço para o Managed Service for Apache Flink

Compatível com perfis de serviço: Sim

Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a uma Serviço da AWS no Guia do IAM usuário.

> **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

Suporte a perfis vinculados a serviços: sim

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. Serviço da AWS O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.

Para obter detalhes sobre como criar ou gerenciar funções vinculadas a serviços, consulte AWS serviços que funcionam com. 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 realizar tarefas usando o AWS Management Console, AWS Command Line Interface (AWS CLI) ou AWS API. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

Para saber como criar uma política IAM baseada em identidade usando esses exemplos de documentos de JSON política, consulte Criação de IAM políticas no Guia do IAMusuário.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Managed Service para Apache Flink, incluindo o formato do ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição do Amazon Managed Service para Apache Flink na Referência de autorização de serviço.

Tópicos

- Melhores práticas de política
- Usando o console do Managed Service for Apache Flink
- Permitir que usuários visualizem suas próprias permissões

Melhores práticas de política

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 seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas

AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte políticas AWS gerenciadas ou políticas AWS gerenciadas para funções de trabalho no Guia IAM do usuário.

- Aplique permissões com privilégios mínimos — Ao definir permissões com IAM políticas, conceda somente as permissões necessárias para realizar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre IAM como usar para aplicar permissões, consulte Políticas e permissões IAM no Guia IAM do usuário.
- Use condições nas IAM políticas para restringir ainda mais o acesso — Você pode adicionar uma condição às suas políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usandoSSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica Serviço da AWS, como AWS CloudFormation. Para obter mais informações, consulte elementos IAM JSON da política: Condição no Guia IAM do usuário.
- Use o IAM Access Analyzer para validar suas IAM políticas e garantir permissões seguras e funcionais — o IAM Access Analyzer valida políticas novas e existentes para que as políticas sigam a linguagem da IAM política (JSON) e as melhores práticas. IAM IAMO Access Analyzer fornece mais de 100 verificações de políticas e recomendações práticas para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação da política do IAM Access Analyzer no Guia do IAM Usuário.
- Exigir autenticação multifatorial (MFA) — Se você tiver um cenário que exija IAM usuários ou um usuário root Conta da AWS, ative MFA para obter segurança adicional. Para exigir MFA quando API as operações são chamadas, adicione MFA condições às suas políticas. Para obter mais informações, consulte Configurando o API acesso MFA protegido no Guia do IAMusuário.

Para obter mais informações sobre as melhores práticas emIAM, consulte as melhores práticas de segurança IAM no Guia IAM do usuário.

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.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para AWS CLI o. ou AWS API o. Em vez disso, permita o acesso somente às ações que correspondam à API operação que eles estão tentando realizar.

Para garantir que usuários e funções ainda possam usar o console do Managed Service for Apache Flink, anexe também o Managed Service for Apache Flink `ConsoleAccess` ou a política `ReadOnly` AWS gerenciada às entidades. Para obter mais informações, consulte Adicionar permissões a um usuário no Guia do IAM usuário.

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permita IAM aos usuários visualizar as políticas embutidas e gerenciadas que estão anexadas à identidade do usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando o AWS CLI ou. 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": "*"
        }
    ]
}
```

Solução de problemas de identidade e acesso para o Amazon Managed Service for Apache Flink

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Managed Service para Apache Flink e. IAM

Tópicos

Não tenho autorização para executar uma ação no Managed Service for Apache Flink

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. 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 uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um IAM usuário chamado `marymajor` tenta usar o console para realizar 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 precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Managed Service for Apache Flink

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode 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 para Apache Flink funciona com IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Fornecer acesso a um IAM usuário em outro Conta da AWS de sua propriedade no Guia do IAM usuário.

- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Fornecer Contas da AWS acesso a terceiros no Guia do IAM usuário.
- Para saber como fornecer acesso por meio da federação de identidades, consulte Fornecendo acesso a usuários autenticados externamente (federação de identidades) no Guia do IAMusuário.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Prevenção do problema do substituto confuso entre serviços

Em AWS, a representação 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 que delegados confusos, AWS fornece ferramentas que ajudam você a proteger seus dados em todos os serviços usando diretores de serviços que receberam acesso aos recursos em sua conta. Esta seção se concentra na prevenção de delegações confusas entre serviços, específica do Managed Service for Apache Flink. No entanto, você pode aprender mais sobre esse tópico na seção O problema confuso do assistente confuso do Guia do IAM usuário.

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 apenas um recurso associado a acessibilidade de serviço. 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 confuso problema do substituto é usar a chave de contexto da condição `aws:SourceArn` global com o recurso completoARN.

Se você não souber a totalidade ARN do recurso ou se estiver especificando vários recursos, use a `aws:SourceArn` chave com caracteres curinga (*) 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 AWS Management Console e abra o IAM console 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 JSON política padrão por uma política que usa uma ou ambas as chaves de contexto de condição `aws:SourceAccount` global. `aws:SourceArn` 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"
            }
         }
      }
   ]
}
```

Validação de conformidade do Amazon Managed Service para Apache Flink

Auditores terceirizados avaliam a segurança e a conformidade do Amazon Managed Service para Apache Flink como parte de vários AWS programas de conformidade. Isso incluiSOC, PCIHIPAA, e outros.

Para obter uma lista de AWS serviços no escopo de programas de conformidade específicos, consulte. Para obter informações gerais, consulte Programas de conformidade da AWS.

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em 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 ouPCI, AWS fornece recursos para ajudar a:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos focados em segurança e conformidade em. AWS
- Arquitetura para HIPAA segurança e conformidade na Amazon Web Services. Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos HIPAA compatíveis.
- AWS Recursos de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Config— Esse AWS serviço avalia se suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Esse AWS serviço fornece uma visão abrangente do seu estado de segurança interno, AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

Alimentado RAMP

O programa de RAMP conformidade do AWS Fed inclui o Managed Service for Apache Flink como um serviço autorizado RAMP pelo Fed. 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 RAMP Segurança do AWS Fed por meio do Fed RAMPPMO, seu gerente de contas de AWS vendas, ou pode baixá-los por meio do AWS Artifact at Artifact AWS .

Para obter mais informações, consulte Fed RAMP.

Resiliência no Amazon Managed Service for Apache Flink

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as Zonas de Disponibilidade, é possível projetar e operar aplicações e bancos de dados que executem 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 datacenter tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, um serviço gerenciado para o Apache Flink oferece vários recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

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 a Amazon. EKS Vários pods do Kubernetes são usados na Amazon EKS para cada AWS região 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 Implemente a tolerância a falhas no Managed Service for Apache Flink.

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 para Apache Flink

Como um serviço gerenciado, o Managed Service for Apache Flink é protegido pelos procedimentos AWS globais de segurança de rede descritos no whitepaper Amazon Web Services: Overview of Security Processes.

Você usa API chamadas AWS publicadas para acessar o Managed Service for Apache Flink pela rede. Todas as API chamadas para o Managed Service for Apache Flink são protegidas via Transport Layer Security (TLS) e autenticadas via. IAM Os clientes devem oferecer suporte à TLS versão 1.2 ou posterior. Os clientes também devem oferecer suporte a pacotes de criptografia com sigilo direto perfeito (), como Ephemeral Diffie-Hellman (PFS) ou Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando uma ID de chave de acesso e uma chave de acesso secreta associada a um IAM principal. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Melhores práticas de segurança para serviços gerenciados para 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 IAM funções para acessar outros serviços da Amazon

Seu aplicativo Managed Service for Apache Flink deve ter credenciais válidas para acessar recursos em outros serviços, como fluxos de dados Kinesis, streams Firehose ou buckets Amazon S3. Você não deve armazenar AWS credenciais 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 uma IAM função para gerenciar credenciais temporárias para que seu 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 Guia IAM do usuário:

- IAMFunções
- Cenários comuns para funções: usuários, aplicativos e serviços

Implemente 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. Você deve implementar a criptografia do lado do servidor em seus recursos dependentes, como streams de dados Kinesis, streams Firehose e buckets Amazon S3. Para obter mais informações sobre como implementar a criptografia do lado do servidor em recursos dependentes, consulte Proteção de dados no Managed Service para Apache Flink.

Use CloudTrail para monitorar API chamadas

O Managed Service for Apache Flink é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função 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 obter mais informações, consulte the section called “Serviço gerenciado de log para chamadas do Apache Flink API com AWS CloudTrail”.

Registro e monitoramento no Amazon Managed Service para 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 sua AWS solução para poder depurar com mais facilidade uma falha multiponto, 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

Login no serviço gerenciado para Apache Flink

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.

Registros de consulta com o CloudWatch Logs 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 em serviço gerenciado para Apache Flink

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 Use 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, essas métricas indicam se o aplicativo está atrasado e precisa ser escalado 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.
- cpuUtilizatione heapMemoryUtilization— 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.

Configurar o registro de aplicativos no Managed Service para Apache Flink

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 oAPI, você também deve criar o grupo de registros e o fluxo de registros do aplicativo e adicionar as permissões que seu aplicativo precisa para gravar no fluxo de registros.

Configurar 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`* é 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 “Controle os níveis de monitoramento de aplicativos”.

Configure o CloudWatch registro usando o CLI

Para adicionar uma opção de CloudWatch registro usando o AWS CLI, você conclui 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.

Crie 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 API o. 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.

Trabalhe com opções de CloudWatch registro de aplicativos

Use as API ações 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 JSON arquivo como entrada para uma API ação, consulteCódigo de exemplo do Managed Service for Apache Flink API.

Adicione 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 *`Amazon Resource Name (ARN)`*

of the CloudWatch Log stream to add to the new application 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>"
 }]
}
```

Adicionar 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 *user input placeholder* com 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>
}
```

Atualizar 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 *`user input placeholder`* com 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>
}
```

Excluir uma opção de CloudWatch registro 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 *`user input placeholder`* com 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>
}
```

Defina o nível de registro 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 “Controle os níveis de monitoramento de aplicativos”.

Defina o nível de registro 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"
}
```

Atualizar o nível de registro 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"
         }
      }
   }
}
```

Adicione permissões para gravar 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 à função AWS Identity and Access Management (IAM) que o Managed Service for Apache Flink assume.

Para obter mais informações sobre o uso de uma IAM função do Managed Service for Apache Flink, consulte. Gerenciamento de identidade e acesso para o Amazon Managed Service for Apache Flink

Política de confiança

Para conceder ao Managed Service for Apache Flink permissões para assumir uma IAM função, você pode anexar a seguinte política de confiança à função de execução do 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 IAM permissões. Forneça os nomes de recursos da Amazon (ARNs) corretos para seu grupo de registros e stream.

```
{
```

```
    "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:*",
            ]
        }
    ]
}
```

Controle os 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 “Implemente o escalonamento de aplicativos no Managed Service para Apache Flink”.
- Operador: as métricas têm como escopo cada operador. Para obter mais informações sobre os operadores, consulte the section called “Transforme dados usando operadores no Managed Service for Apache Flink”.
- Paralelismo: as métricas têm como escopo o paralelismo de aplicativos. Você só pode definir esse nível de métrica usando o MonitoringConfigurationUpdateparâmetro do UpdateApplicationAPI. Você não pode definir esse nível de métricas usando o console. Para obter mais informações sobre paralelismo, consulte the section called “Implemente o escalonamento de aplicativos no Managed Service para Apache Flink”.

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.

Aplique as melhores práticas de registro

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.

Executar solução de problemas de registro

Se os registros do aplicativo não estiverem sendo gravados no fluxo de logs, verifique o seguinte:

- Verifique se a IAM função e as políticas do seu aplicativo estão corretas. Se a política do seu aplicativo precisa das seguintes permissões para acessar seu fluxo de logs:
  - `logs:PutLogEvents`
  - `logs:DescribeLogGroups`
  - `logs:DescribeLogStreams`

  Para obter mais informações, consulte the section called “Adicione permissões para gravar 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, consulte???.

Use o CloudWatch Logs Insights

Depois de habilitar o CloudWatch login no seu aplicativo, você pode usar o CloudWatch Logs Insights para analisar os registros do seu aplicativo. Para obter mais informações, consulte the section called “Analise registros com o CloudWatch Logs Insights”.

Analise 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 um exemplo de consulta

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 Amazon Route 53 ou AmazonVPC, provavelmente já configurou registros desses serviços para acessar CloudWatch 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.

Analise exemplos de consultas

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*), ID da conta (*012345678901*) e nome do aplicativo (*YourApplication*) nos exemplos de consulta a seguir com a região do seu aplicativo e o ID da sua conta.

Este tópico contém as seguintes seções:

- Analise as operações: distribuição de tarefas
- Analise as operações: mudança no paralelismo
- Analisar erros: acesso negado
- Analisar erros: fonte ou coletor não encontrados
- Analise 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 Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

```
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 a seguir do CloudWatch Logs Insights 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 “Use o escalonamento automático no Managed Service para Apache Flink”.

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Analisar erros: acesso negado

A consulta a seguir do CloudWatch Logs Insights retorna `Access Denied` registros.

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Analisar erros: fonte ou coletor não encontrados

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:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

Analise 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:kinesisanalytics:us-
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:kinesisanalytics:us-
west-2:012345678901:application\/YourApplication/
| filter @message like /switched from RUNNING to FAILED/
| sort @timestamp desc
```

Métricas e dimensões no Managed Service para Apache Flink

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 da aplicação

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
`backPressuredTimeMsPerSecond*`	Milissegundos	O tempo (em milissegundos) em que essa tarefa	Tarefa, operador, paralelismo	*Disponível para aplicativos Managed Service para

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
		ou operador é contrapressionado por segundo.		Apache Flink executando somente a versão 1.13 do Flink. Essas métricas podem ser 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 CPU utilização 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 CPU utilização mínima, média e máxima em seu aplicativo. A `CPUUtilization` métrica considera apenas o CPU uso do TaskManager JVM processo em execução dentro do contêiner.	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
`containerCPUUtilization`	Porcentagem	Porcentagem geral de CPU utilização em contêineres do gerenciador de tarefas no cluster de aplicativos 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: CPUTempo total (em segundos) consumido pelo contêiner * 100/ CPU Limite do contêiner (em CPUs / segundos) A `CPUUtilization` métrica considera apenas o CPU uso do TaskManager JVM processo em execução dentro do contêiner. Há outros componentes funcionando do fora do JVM mesmo contêiner. A `containerCPUUtiliz`	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				`ation` métrica fornece uma visão mais completa, incluindo todos os processos em termos de CPU exaustão 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 TaskManager JVM ou	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
				memória gerenciada (uso de memória externa JVM para processos nativos, como 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 Heap Memory Usage of 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 `ManagedMemoryUtilzations` métrica considera apenas métricas de memória específicas, como memória gerenciad

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
				a (uso de memória externa JVM para processos nativos, como 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 SUM estatística 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 tira 4 instantâneos métricos por minuto, a seguinte matemática métrica deve	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				ser usada: m1/4, onde m1 é a SUM estatística de 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 SUM estatística 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 tira 4 instantâneos métricos por minuto, a seguinte matemática métrica deve	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				ser usada: m1/4, onde m1 é a SUM estatística de 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 SUM estatística 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 tira 4 instantâneos métricos por minuto, a seguinte matemática métrica deve

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				ser usada: m1/4, onde m1 é a SUM estatística de 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 SUM estatística 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 tira 4 instantâneos métricos por minuto, a seguinte matemática métrica deve	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				ser usada: m1/4, onde m1 é a SUM estatística de 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 SUM estatística 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 tira 4 instantâneos métricos por minuto, a seguinte matemática métrica deve	

Métrica	Unidade	Descrição	Nível	Observações sobre o uso	
				ser usada: m1/4, onde m1 é a SUM estatística de 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 por segundo.	
`oldGenera tionGCCou nt`	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.

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
KPUs*	Contagem	O número total de KPUs usados pelo aplicativo.	Aplicativo	*Essa métrica recebe uma amostra por período de cobrança (uma hora). Para visualizar o número de KPUs ao longo do tempo, use MAX ou AVG durante um período de pelo menos uma (1) hora. A KPU contagem inclui `orchestration` KPUo. Para obter mais informações, consulte Preços do Managed Service for Apache Flink.

Métricas do conector 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 chamada para `getRecords` .	Aplicação (para Stream), Paralelismo (para) ShardId	

Métricas MSK do conector Amazon

AWS emite todos os registros para a 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 não afeta a integridade dos deslocamentos de partição do ponto de

Métrica	Unidade	Descrição	Nível	Observações sobre o uso
				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 Studio, AWS emite as seguintes métricas no nível do aplicativo:`KPUs`,`cpuUtilization`,`heapMemoryUtilization`, `oldGenerationGCTime`e `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 CPU utilização 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`

Exibir 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 o AWS CLI

- Em um prompt de comando, use o seguinte comando.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Defina 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 CLI o. 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.

Use métricas personalizadas com o Amazon Managed Service para 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
- Veja exemplos para criar uma classe de mapeamento
- Exibir 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.

Veja exemplos para criar uma classe de mapeamento

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 `DataStream` objeto.

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.

Exibir métricas personalizadas

As métricas personalizadas do seu aplicativo aparecem no console de CloudWatch métricas no AWS/KinesisAnalyticspainel, no grupo de métricas do aplicativo.

Use 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 ação é enviar uma notificação para um tópico do Amazon Simple Notification Service (AmazonSNS).

Para obter mais informações sobre CloudWatch alarmes, consulte Usando CloudWatch alarmes da Amazon.

Revise os 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
`downtime > 0`	Média	0	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. Recomendado para todas as aplicações. A `Downtime` métrica mede a duração de uma interrupção. Um tempo de inatividade maior que zero indica que o aplicativo falhou. Para solução de problemas, consulte O aplicativo está sendo reiniciado.
`RATE (numberOfFailedCheckpoints) > 0`	Média	0	Essa métrica conta o número de pontos de verificação com falha desde o início do aplicativo. Dependend

Expressão da métrica	Estatística	Limite	Descrição
			o do aplicativo, pode ser tolerável que os pontos de verificaç ão falhem de vez em quando. Mas, se os pontos de verificação falharem regularme nte, é provável que o aplicativo não esteja íntegro e precise de mais atenção. Recomendamos o monitoramento RATE (numberOfFailedpon tos de verificação) para alertar sobre o gradiente e não sobre valores absolutos. Recomendado para todas as aplicaçõe s. Use essa métrica para monitorar a integridade do aplicativo e verificar o progresso. O aplicativ o salva os dados do estado nos pontos de verificação quando estão íntegros. O ponto de verificação pode falhar devido a tempos limite se o aplicativo não estiver progredindo no processamento

Expressão da métrica	Estatística	Limite	Descrição
			dos dados de entrada. Para solução de problemas, consulteO ponto de verificação está atingindo o tempo limite.
`Operator.numRecordsOutPerSecond` < limite	Média	O número mínimo de registros emitidos pelo aplicativo em condições normais.	Recomendado para todas as aplicações. Ficar abaixo desse limite pode indicar que o aplicativo não está fazendo o progresso esperado nos dados de entrada. Para solução de problemas, consulteA taxa de transferência é muito lenta.

Expressão da métrica	Estatística	Limite	Descrição
`records_lag_max|millisbehindLatest` > limite	Máximo	A latência máxima esperada em condições normais.	Se o aplicativo estiver consumindo do Kinesis ou do Kafka, essas métricas indicam se o aplicativo está ficando para trás e precisa ser escalado 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. Recomendado para todas as aplicações. Use a `records_lag_max` métrica para uma fonte do Kafka ou `millisbehindLatest` para uma fonte de stream do Kinesis. Superar esse limite pode indicar que o aplicativo não está fazendo o progresso esperado nos dados de entrada. Para solução de

Expressão da métrica	Estatística	Limite	Descrição
			problemas, consulte A taxa de transferência é muito lenta.

Expressão da métrica	Estatística	Limite	Descrição
`lastCheckpointDuration` > limite	Máximo	A duração máxima esperada do ponto de verificação em condições normais.	Monitora a quantidade de dados armazenados no estado e quanto tempo leva para passar por 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)`. Se o `lastCheck`

Expressão da métrica	Estatística	Limite	Descrição
			`pointDuration` aumento continuar , ultrapassar esse limite pode indicar que o aplicativo não está fazendo o progresso esperado nos dados de entrada ou que há problemas com a integridade do aplicativo, como contrapressão. Para solução de problemas , consulteCrescimento estadual ilimitado.

Expressão da métrica	Estatística	Limite	Descrição
`lastCheckpointSize` > limite	Máximo	O tamanho máximo esperado do ponto de verificação em condições normais.	Monitora a quantidade de dados armazenados no estado e quanto tempo leva para passar por 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)`. Se o `lastCheck`

Expressão da métrica	Estatística	Limite	Descrição
			`pointSize` aumento continuar , ultrapassar esse limite pode indicar que o aplicativo está acumulando dados de estado. Se os dados de estado ficarem muito grandes, o aplicativo poderá ficar sem memória ao se recuperar de um ponto de verificação, ou a recuperação de um ponto de verificação poderá levar muito tempo. Para solução de problemas , consulteCrescimento estadual ilimitado.

Expressão da métrica	Estatística	Limite	Descrição
`heapMemoryUtilization` > limite	Máximo	Isso fornece uma boa indicação da utilização geral dos recursos do aplicativo e pode ser usado para escalabilidade proativa, a menos que o aplicativo esteja vinculado à E/S. O `heapMemoryUtilization` tamanho máximo esperado em condições normais, com um valor recomendado de 90 por cento.	Você pode usar essa métrica para monitorar a utilização máxima da memória dos gerenciadores de tarefas em todo o aplicativo. Se o aplicativo atingir esse limite, você precisará provisionar mais recursos. Você faz isso ativando o escalonamento automático ou aumentando o paralelismo do aplicativo. Para obter mais informações sobre o aumento de recursos, consulte Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

Expressão da métrica	Estatística	Limite	Descrição
`cpuUtilization` > limite	Máximo	Isso fornece uma boa indicação da utilização geral dos recursos do aplicativo e pode ser usado para escalabilidade proativa, a menos que o aplicativo esteja vinculado à E/S. O `cpuUtilization` tamanho máximo esperado em condições normais, com um valor recomendado de 80 por cento.	Você pode usar essa métrica para monitorar a CPU utilização máxima dos gerenciadores de tarefas em todo o aplicativo. Se o aplicativo atingir esse limite, você precisará provisionar mais recursos. Faça isso ativando a escalabilidade automática ou aumentando o paralelismo do aplicativo. Para obter mais informações sobre o aumento de recursos, consulte Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

Expressão da métrica	Estatística	Limite	Descrição
`threadsCount` > limite	Máximo	O `threadsCount` tamanho máximo esperado em condições normais.	Você pode usar essa métrica para observar vazamentos de tópicos nos gerenciadores de tarefas em todo o aplicativo. Se essa métrica atingir esse limite, verifique se há threads criados sem serem fechados no código do aplicativo.
`(oldGarbageCollectionTime * 100)/60_000 over 1 min period')` > limite	Máximo	A `oldGarbageCollectionTime` duração máxima esperada. Recomendamos definir um limite de forma que o tempo normal de coleta de lixo seja 60% do limite especificado, mas o limite correto para seu aplicativo variará.	Se essa métrica estiver aumentando continuamente, isso pode indicar que há um vazamento de memória nos gerenciadores de tarefas em todo o aplicativo.
`RATE(oldGarbageCollectionCount)` > limite	Máximo	O máximo esperado `oldGarbageCollectionCount` em condições normais. O limite correto para sua inscrição variará.	Se essa métrica estiver aumentando continuamente, isso pode indicar que há um vazamento de memória nos gerenciadores de tarefas em todo o aplicativo.

Expressão da métrica	Estatística	Limite	Descrição
`Operator.currentOutputWatermark - Operator.currentInputWatermark` > limite	Mínimo	O incremento mínimo esperado da marca d'água em condições normais. O limite correto para sua inscrição variará.	Se essa métrica estiver aumentando continuamente, isso pode indicar que o aplicativo está processando eventos cada vez mais antigos ou que uma subtarefa inicial não envia uma marca d'água há um tempo cada vez mais longo.

Grave mensagens personalizadas no CloudWatch Logs

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
- Grave CloudWatch nos registros usando SLF4J

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"
}
```

Grave CloudWatch nos 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"
}
```

Serviço gerenciado de log para chamadas do Apache Flink API com AWS CloudTrail

O Managed Service for Apache Flink é integrado com AWS 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 API chamadas para o Managed Service for Apache Flink como eventos. As chamadas capturadas incluem chamadas do serviço gerenciado para o console Apache Flink e chamadas de código para o serviço gerenciado para operações do Apache Flink.

API 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.

Informações do Managed Service for 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. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em sua AWS conta, 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, a trilha se aplica a todas as AWS regiões. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. 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 as informações a seguir.

- Visão Geral para Criar uma Trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando as SNS notificações da Amazon 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 do Managed Service for Apache Flink. API 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 root ou AWS Identity and Access Management (IAM).
- 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 AWS serviço.

Para obter mais informações, consulte o CloudTrail userIdentityElemento.

Entenda as entradas do 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 API chamadas públicas, 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"
        }
    ]
}
```

Ajuste o desempenho no Amazon Managed Service para Apache Flink

Este tópico descreve técnicas para monitorar e melhorar o desempenho do seu aplicativo Managed Service for Apache Flink.

Tópicos

Solucionar 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.

Entenda 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 apenas dados 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

CloudWatch níveis de monitoramento

Verifique se os níveis de CloudWatch monitoramento não estão definidos para 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 Controle os níveis de monitoramento de aplicativos.

CPUMétrica de aplicação

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 escalonamento automático estiver ativado, o aplicativo aloca mais recursos se o CPU uso for superior a 75 por cento por 15 minutos. Para obter mais informações sobre escalonamento, consulte a seção Gerenciar a escalabilidade de forma adequada a seguir, e Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

> **Note**
>
> Um aplicativo só será escalado automaticamente em resposta ao CPU uso. 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 ação. UpdateApplication

O máximo 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 de operadores a seguir.

Registro em log de aplicações

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 “Analise registros com o CloudWatch Logs Insights”.

Paralelismo de operadores

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 de aplicação

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.

Use as 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. Suas inscrições KPUs aumentarão automaticamente se os seguintes critérios forem atendidos:

- O ajuste de escala automático está ativado para o aplicativo.
- CPUo uso permanece acima de 75 por cento por 15 minutos.

Se o escalonamento automático estiver ativado, mas o CPU uso não permanecer nesse limite, o aplicativo não será ampliado. KPUs Se você tiver um aumento no CPU uso 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 Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

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 Use propriedades de tempo de execução no Managed Service para Apache Flink, 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, consulteAnalise os recursos do aplicativo Managed Service for Apache Flink.

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
- DataStream Transformações

Monitore o uso de recursos de dependência externa

Se houver um gargalo de desempenho em um destino (como Kinesis Streams, Firehose, DynamoDB ou Service), seu aplicativo sofrerá contrapressão. OpenSearch 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 CloudWatch nos registros dos 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 Standalone na documentação do Apache Flink.

Monitore o desempenho

Esta seção descreve as ferramentas para monitorar o desempenho de um aplicativo.

Monitore o desempenho usando CloudWatch métricas

Você monitora o uso de recursos, a taxa de transferência, o ponto de verificação e o tempo de inatividade do seu aplicativo usando métricas. CloudWatch Para obter informações sobre o uso de CloudWatch métricas com seu aplicativo Managed Service for Apache Flink, consulte. ???

Monitore o desempenho usando CloudWatch registros e alarmes

Você monitora condições de erro que podem 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 CloudWatch alarmes para criar notificações sobre problemas de desempenho, como uso de recursos ou métricas de ponto de verificação acima de um limite seguro ou alterações inesperadas no status do aplicativo.

Para obter informações sobre a criação de CloudWatch alarmes para um aplicativo Managed Service for Apache Flink, consulte. ???

Serviço gerenciado para cota de notebooks Apache Flink e Studio

> **Note**
>
> As versões 1.6, 1.8 e 1.11 do Apache Flink não são suportadas pela comunidade Apache Flink há mais de três anos. Planejamos descontinuar essas versões no Amazon Managed Service para Apache Flink em 5 de novembro de 2024. A partir dessa data, você não poderá criar novos aplicativos para essas versões do Flink. Você pode continuar executando os aplicativos existentes no momento. Você pode atualizar seus aplicativos de forma estável usando o recurso de atualizações de versão in-loco no Amazon Managed Service para Apache Flink. Para obter mais informações, consulte. Use atualizações de versão in-loco para o Apache Flink

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.

- O número 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 KPU limite precisa ser aplicado.

  Com o Managed Service for Apache Flink, sua AWS conta é cobrada pelos recursos alocados, em vez dos recursos que seu aplicativo usa. É cobrada uma taxa horária com base no número máximo usado para executar seu aplicativo de KPUs processamento de stream. Um único KPU fornece 1 v CPU e 4 GiB de memória. Para cada umKPU, o serviço também provisiona 50 GiB de armazenamento de aplicativos em execução.

- É possível criar até 1.000 Gerencie backups de aplicativos usando instantâneos do Managed Service for Apache Flink por aplicativo.

- Você pode atribuir até 50 tags por aplicativo.

- O tamanho máximo de um JAR arquivo de 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

Gerencie tarefas de 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 o. UpdateApplicationMaintenanceConfigurationAPI

Região	Janela de tempo de manutenção
AWS GovCloud (Oeste dos EUA)	06:00 — 14:00 UTC
AWS GovCloud (Leste dos EUA)	03:00 — 11:00 UTC
Leste dos EUA (Norte da Virgínia)	03:00 — 11:00 UTC
Leste dos EUA (Ohio)	03:00 — 11:00 UTC
Oeste dos EUA (N. da Califórnia)	06:00 — 14:00 UTC
Oeste dos EUA (Oregon)	06:00 — 14:00 UTC
Ásia-Pacífico (Hong Kong)	13:00 — 21:00 UTC
Ásia-Pacífico (Mumbai)	16:30 — 00:30 UTC
Ásia-Pacífico (Hyderabad)	16:30 — 00:30 UTC
Ásia-Pacífico (Seul)	13:00 — 21:00 UTC

Região	Janela de tempo de manutenção
Ásia-Pacífico (Singapura)	14:00 — 22:00 UTC
Ásia-Pacífico (Sydney)	12:00 — 20:00 UTC
Ásia-Pacífico (Jacarta)	15:00 — 23:00 UTC
Ásia-Pacífico (Tóquio)	13:00 — 21:00 UTC
Canadá (Central)	03:00 — 11:00 UTC
China (Pequim)	13:00 — 21:00 UTC
China (Ningxia)	13:00 — 21:00 UTC
Europa (Frankfurt)	06:00 — 14:00 UTC
Europa (Zurique)	20:00 — 04:00 UTC
Europa (Irlanda)	22:00 — 06:00 UTC
Europa (Londres)	22:00 — 06:00 UTC
Europa (Estocolmo)	23:00 — 07:00 UTC
Europa (Milão)	21:00 — 05:00 UTC
Europa (Espanha)	21:00 — 05:00 UTC
África (Cidade do Cabo)	20:00 — 04:00 UTC
Europa (Irlanda)	22:00 — 06:00 UTC
Europa (Londres)	23:00 — 07:00 UTC
Europa (Paris)	23:00 — 07:00 UTC
Europa (Estocolmo)	23:00 — 07:00 UTC
Oriente Médio (Barém)	13:00 — 21:00 UTC

Região	Janela de tempo de manutenção
Oriente Médio (UAE)	18:00 — 02:00 UTC
América do Sul (São Paulo)	19:00 — 03:00 UTC
Israel (Tel Aviv)	20:00 — 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 um operador explícito e consistente IDs para os operadores do gráfico de tarefas do Flink. Se não for definido explicitamente, o Flink gera automaticamente um ID para os operadores. Isso ocorre porque o Flink usa esses operadores IDs para identificar exclusivamente os operadores em um gráfico de tarefas e os usa para armazenar o estado de cada operador em um ponto de salvamento.

O problema de incompatibilidade de ID do operador ocorre quando o Flink não encontra um mapeamento 1:1 entre o operador IDs de um gráfico de tarefas e o operador IDs definido em um ponto de salvamento. Isso acontece quando operadores consistentes explícitos não IDs são definidos e o Flink gera automaticamente um operador IDs que pode não ser consistente com cada criação de gráfico de tarefas. A probabilidade de os aplicativos enfrentarem esse problema é alta durante as operações de manutenção. Para evitar isso, recomendamos que os clientes definam UUID para todos os operadores no código Flink. Para obter mais informações, consulte o tópico Definir a UUID para todos os operadores em Preparação para produção.

Identifique quando ocorreu a manutenção em seu aplicativo

Você pode descobrir se o Managed Service for Apache Flink realizou uma ação de manutenção em seu aplicativo usando o. `ListApplicationOperations` API

Veja a seguir um exemplo de solicitação `ListApplicationOperations` que pode ajudá-lo a filtrar a lista para manutenção no aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "operation": "ApplicationMaintenance"
```

```
}
```

Obtenha prontidão de produção para seu serviço gerenciado para aplicativos Apache Flink

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 a carga de seus aplicativos

Alguns problemas com aplicativos só se manifestam sob carga pesada. Vimos casos em que os aplicativos pareciam saudáveis, mas 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. Quando esse problema é corrigido, há um acúmulo de dados não processados que se acumulam, o que pode esgotar completamente os recursos disponíveis. A carga pode então amplificar bugs ou problemas de desempenho que não haviam surgido antes.

Portanto, é essencial que você execute 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 dos tempos. Por exemplo, use uma posição inicial de `TRIM_HORIZON` no caso de um stream de dados do Kinesis.

Defina o 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 você planeja escalar seu aplicativo > 128 paralelismo, você deve definir explicitamente o paralelismo máximo.

Você pode definir o paralelismo máximo no nível do aplicativo, com um único `env.setMaxParallelism(x)` operador. A menos que especificado de outra forma, todos os operadores herdam o paralelismo máximo do aplicativo.

Para obter mais informações, consulte Definindo o paralelismo máximo na documentação do Apache Flink.

Defina um UUID para todos os operadores

A UUID é usado na operação na qual o Flink mapeia um ponto de salvamento de volta para um operador individual. Definir um específico UUID 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.

Mantenha as melhores práticas de serviço gerenciado para aplicativos 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 Gerencie tarefas de manutenção do Managed Service for Apache Flink.

- 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 ???.

Para obter mais informações sobre a implementação da tolerância a falhas, consulte Implemente a tolerância a falhas no Managed Service for Apache Flink.

Versões de conectores incompatíveis

A partir da versão 1.15 ou posterior do Apache Flink, o Managed Service for Apache Flink impede automaticamente que os aplicativos sejam iniciados ou atualizados se estiverem usando versões incompatíveis do conector Kinesis agrupadas no aplicativo. JARs Ao atualizar para o Managed Service for Apache Flink versão 1.15 ou posterior, verifique se você está usando o conector Kinesis mais recente. Isso quer dizer, qualquer versão igual ou mais recente do que a versão 1.15.2. Todas as outras versões não são suportadas pelo Managed Service for Apache Flink porque elas podem causar problemas de consistência ou falhas com o recurso Stop with Savepoint, impedindo operações limpas de parada/atualização. Para saber mais sobre a compatibilidade de conectores nas versões do Amazon Managed Service para Apache Flink, consulte Conectores Apache Flink.

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 Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

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 o using. DataStream API `.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 ter mais informações, consulte ???

Para obter mais informações sobre o registro em log, consulte ???.

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 JAR arquivo do aplicativo e as 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 JAR arquivo 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 a autenticação seja necessária para um serviço que não está nativamente integrado comIAM, por exemplo, um banco de dados que requer um nome de usuário e senha para autenticação, considere armazenar AWS segredos no Secrets Manager.

Muitos serviços AWS nativos 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 seguindo uma função keyBy (...) ou partição (...). 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:
```

```
 4962789433850315183450815791266608495756527394990168485 0,}}'} 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:
 4962789433845855034411109666638301352101997722688972398 6,}}'}, 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:
 4962789433845855034411109666638301352101997722688972398 6,}}'} 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:
 4962789433843624959891256604324147780274732886538374355 4,}}'}, 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:
 4962789433841394885371403542009994208447468050387776312 2,}}'}, 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:
 4962789433868155779609640289779837070374646084194952830 6,}}'}, 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:
```

```
 496278943386592570508978722746568349854738124804435478 74,}}'}, 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:
 191408831393027885698148216680369618944,EndingHashKey:
 212676479325586539664609129644855132159},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:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
```

```
 49627894338614655560500811028373763548928515757431587010,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 49627894338525452579706688535807620675837922311407665282,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132 16,EndingHashKey:
 425352958651173079329218259289710264 31},SequenceNumberRange: {StartingSequenceNumber:
 49627894338369347363316974173816870647929383780865802258,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 49627894338592354815302280405232227830655867395925606578,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 212676479325586539664609129644855132 15},SequenceNumberRange: {StartingSequenceNumber:
 49627894338347046618118443550675334929656735419359821826,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 425352958651173079329218259289710264 32,EndingHashKey:
 638029437976759618993827388934565396 47},SequenceNumberRange: {StartingSequenceNumber:
 49627894338391648108515504796958406366202032142371782690,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
```

```
 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:
 496278943383916481085155047969584063662020321423717826 90,}}'} 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:
 19140883139302788569814821668036961894 4,EndingHashKey:
 21267647932558653966460912964485513215 9},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457 14,}}'} 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:
 21267647932558653966460912964485513216 0,EndingHashKey:
 23394412725814519363107004260934064537 5},SequenceNumberRange: {StartingSequenceNumber:
 496278943385700540701037497820906921123832190344196261 46,}}'} 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:
 31901471898837980949691369446728269824 0,EndingHashKey:
 34028236692093846346337460743176821145 5},SequenceNumberRange: {StartingSequenceNumber:
 496278943386815577960964028977983707037464608419495283 06,}}'} 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:
 29774707105582115553045278150279718502 4,EndingHashKey:
 31901471898837980949691369446728269823 9},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478 74,}}'} 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 um operador explícito e consistente IDs para os operadores do gráfico de tarefas do Flink. Se não for definido explicitamente, o Flink gera automaticamente um ID para os operadores. Isso ocorre porque o Flink usa esses operadores IDs para identificar exclusivamente os operadores em um gráfico de tarefas e os usa para armazenar o estado de cada operador em um ponto de salvamento.

O problema de incompatibilidade de ID do operador ocorre quando o Flink não encontra um mapeamento 1:1 entre o operador IDs de um gráfico de tarefas e o operador IDs definido em

um ponto de salvamento. Isso acontece quando operadores consistentes explícitos não IDs são definidos e o Flink gera automaticamente um operador IDs que pode não ser consistente com cada criação de gráfico de tarefas. A probabilidade de os aplicativos enfrentarem esse problema é alta durante as operações de manutenção. Para evitar isso, recomendamos que os clientes definam UUID para todos os operadores no código flink. Para obter mais informações, consulte o tópico Definir a UUID para todos os operadores em Preparação para produção.

Adicionar ServiceResourceTransformer ao plugin Maven Shade

O Flink usa as interfaces de provedor de serviços (SPI) do Java para carregar componentes como conectores e formatos. O uso de várias dependências do Flink SPI pode causar conflitos no uber-jar e comportamentos inesperados do aplicativo. É recomendável adicionar o plugin ServiceResourceTransformerde sombra Maven, 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 do 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 em 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.

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 pelo Managed Service para aplicativos Apache Flink em código, e outros não podem ser alterados.

Use os links desta seção para saber mais sobre as configurações do Apache flink e quais são modificáveis.

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, para 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 pode 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.

Backend estadual

O Managed Service for Apache Flink armazena dados transitórios em um estado de back-end. O serviço gerenciado para Apache Flink usa o R Backend. ocksDBState 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

Para obter mais informações sobre back-ends estaduais, 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 usando CheckpointConfiguration. Você deve definir como `CheckpointConfiguration.ConfigurationType` `CUSTOM` para que o Managed Service for Apache Flink use valores de checkpoint modificados.

Configuração	Pode ser modificada?	Como	Valor padrão
CheckpointingEnabled	Modificável	Criar aplicativo Atualizar um aplicativo AWS CloudFormation	Verdadeiro
CheckpointInterval	Modificável	Criar aplicativo Atualizar um aplicativo AWS CloudFormation	60000

Configuração	Pode ser modificada?	Como	Valor padrão
MinPauseBetweenCheckpoints	Modificável	Criar aplicativo Atualizar um aplicativo AWS CloudFormation	5000
Pontos de verificação não alinhados	Modificável	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
Localização do ponto de verificação e do ponto de salvamento	Não modificável	N/D	Armazenamos dados duráveis de pontos de verificação e pontos de salvamento em um bucket S3 de propriedade do serviço.

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 AllowNonRestoredStateparâmetro do aplicativo FlinkRunConfigurationpara`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 pilha

O Managed Service for Apache Flink aloca cada 3 KPU GiB de JVM heap e reserva 1 GiB para alocações de código nativo. Para obter informações sobre como aumentar a capacidade do seu aplicativo, consulte the section called “Implemente o escalonamento de aplicativos no Managed Service para Apache Flink”.

Para obter mais informações sobre tamanhos de JVM pilha, 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.

Estratégia de reinício

A partir do Flink 1.19 e versões posteriores, usamos a estratégia de `exponential-delay` reinicialização por padrão. Todas as versões anteriores usam a estratégia de `fixed-delay` reinicialização por padrão.

```
restart-strategy:
```

```
restart-strategy.fixed-delay.delay:
```

```
restart-strategy.exponential-delay.backoff-muliplier:
```

```
restart-strategy.exponential-delay.initial-backoff:
```

```
restart-strategy.exponential-delay.jitter-factor:
```

```
restart-strategy.exponential-delay.reset-backoff-threshold:
```

Pontos de verificação e back-ends estaduais

```
state.backend:
```

```
state.backend.fs.memory-threshold:
```

```
state.backend.incremental:
```

Pontos de verificação

```
execution.checkpointing.unaligned:
```

```
execution.checkpointing.interval-during-backlog:
```

Métricas nativas do RocksDB

As métricas nativas do RocksDB não são enviadas para o. CloudWatch Uma vez ativadas, essas métricas podem ser acessadas no painel do Flink ou no Flink REST API com ferramentas personalizadas.

O Managed Service para Apache Flink permite que os clientes acessem o Flink mais recente RESTAPI(ou a versão compatível que você está usando) no modo somente leitura usando o. CreateApplicationPresignedUrlAPI Isso API é usado pelo próprio painel do Flink, mas também pode ser usado 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 back-ends de estado

```
state.storage.fs.memory-threshold:
```

TaskManager Opções completas

```
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 sistema de arquivos

```
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:`

Metrics

`metrics.latency.interval:`

Opções avançadas para o REST endpoint e o cliente

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Opções avançadas SSL de segurança

`security.ssl.internal.handshake-timeout:`

Opções avançadas de agendamento

`slot.request.timeout:`

Opções avançadas para a interface web do Flink

`web.timeout:`

Exibir 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.

Configure o Managed Service para o Apache Flink para acessar recursos em uma Amazon VPC

Você pode configurar um serviço gerenciado para o aplicativo Apache Flink para se conectar a sub-redes privadas em uma nuvem privada virtual (VPC) em sua conta. Use a Amazon Virtual Private Cloud (AmazonVPC) para criar uma rede privada para recursos como bancos de dados, instâncias de cache ou serviços internos. Conecte seu aplicativo ao VPC para acessar recursos privados durante a execução.

Este tópico contém as seguintes seções:

VPCConceitos da Amazon

A Amazon VPC é a camada de rede da AmazonEC2. Se você é novo na AmazonEC2, consulte O que é a AmazonEC2? no Guia do EC2 usuário da Amazon para instâncias Linux para obter uma breve visão geral.

A seguir estão os conceitos-chave paraVPCs:

- Uma nuvem privada virtual (VPC) é uma rede virtual dedicada à sua AWS conta.
- Uma sub-rede é um intervalo de endereços IP em seuVPC.
- 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 de internet é um VPC componente dimensionado horizontalmente, redundante e altamente disponível que permite a comunicação entre suas instâncias e a Internet. VPC Portanto, não impõe nenhum risco de disponibilidade ou restrições de largura de banda no tráfego da rede.
- Um VPCendpoint permite que você o conecte de forma privada VPC aos AWS serviços suportados e aos serviços de VPC endpoint fornecidos por, PrivateLink sem a necessidade de um gateway de

internet, NAT dispositivo, VPN conexão ou conexão. AWS Direct Connect As suas instâncias VPC não exigem endereços IP públicos para se comunicar com os recursos no serviço. O tráfego entre o seu VPC e o outro serviço não sai da rede Amazon.

Para obter mais informações sobre o VPC serviço da Amazon, consulte o Guia do usuário da Amazon Virtual Private Cloud.

O Managed Service for Apache Flink cria interfaces de rede elásticas em uma das sub-redes fornecidas em sua VPC configuração para o aplicativo. O número de interfaces de rede elástica criadas em suas VPC sub-redes pode variar, dependendo do paralelismo e do paralelismo por aplicativo. KPU Para obter mais informações sobre a escalabilidade do aplicativo, consulte Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.

> **Note**
>
> VPCas configurações não são suportadas por SQL aplicativos.

> **Note**
>
> O serviço Managed Service for Apache Flink gerencia o estado do ponto de verificação e do snapshot para aplicativos que têm uma configuração. VPC

VPCpermissões do aplicativo

Esta seção descreve as políticas de permissão que seu aplicativo precisará para funcionar com o seuVPC. 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 umVPC. Para usar essa política de permissão, adicione-a à função de execução do seu aplicativo.

Adicione uma 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 CloudWatch Logs ou AmazonVPC), 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 serviço VPC gerenciado conectado para o aplicativo Apache Flink

Por padrão, quando você conecta um aplicativo Managed Service for Apache Flink a um aplicativo VPC em sua conta, ele não tem acesso à Internet, a menos que 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.
- Eles VPC devem conter um NAT gateway ou instância em uma sub-rede pública.
- Deve existir uma rota para o tráfego de saída das sub-redes privadas para o NAT gateway em uma sub-rede pública.

> **Note**
>
> Vários serviços oferecem VPCendpoints. Você pode usar VPC endpoints para se conectar aos serviços da Amazon de dentro e 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 NAT gateway (nat-...) ou NAT instância (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 umaNAT) e uma ou mais sub-redes privadas, faça o seguinte para identificar suas sub-redes pública e privada:

- No VPC console, no painel de navegação, escolha 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 /flink https://console.aws.amazon.com
- 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 VPCConectividade, escolha a VPC para associar ao seu aplicativo. Escolha as sub-redes e o grupo de segurança associados à sua VPC que você deseja que o aplicativo use para acessar VPC os recursos.
- Selecione Atualizar.

Informações relacionadas

Criando um VPC com sub-redes públicas e privadas

NATnoções básicas sobre gateway

Use o serviço gerenciado para o Apache Flink VPC API

Use o seguinte serviço gerenciado para API operações do Apache Flink VPCs para gerenciar seu aplicativo. Para obter informações sobre como usar o Managed Service para Apache FlinkAPI, consulte. Código de exemplo de API

Criar aplicativo

Use a CreateApplicationação para adicionar uma VPC configuração ao seu aplicativo durante a criação.

O exemplo a seguir de código de solicitação para a `CreateApplication` ação inclui uma VPC configuração 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 AddApplicationVpcConfigurationação para adicionar uma VPC configuração ao seu aplicativo após sua criação.

O exemplo a seguir de código de solicitação para a `AddApplicationVpcConfiguration` ação adiciona uma VPC configuração a um aplicativo existente:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

DeleteApplicationVpcConfiguration

Use a DeleteApplicationVpcConfigurationação para remover uma VPC configuração do seu aplicativo.

O exemplo a seguir de código de solicitação para a `AddApplicationVpcConfiguration` ação remove uma VPC configuração existente de um aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

Atualizar aplicativo

Use a UpdateApplicationação para atualizar todas as VPC configurações de um aplicativo de uma só vez.

O exemplo a seguir de código de solicitação para a `UpdateApplication` ação atualiza todas VPC as configurações de um aplicativo:

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

Exemplo: Use a VPC para acessar dados em um MSK cluster da Amazon

Para obter um tutorial completo sobre como acessar dados de um MSK cluster da Amazon em umVPC, consulteReplicação do MSK.

Solucionar problemas de serviço gerenciado para Apache Flink

Os tópicos a seguir podem ajudá-lo a solucionar problemas que você pode encontrar com o Amazon Managed Service para Apache Flink.

Escolha o tópico apropriado para analisar as soluções.

Tópicos

Solução de problemas de desenvolvimento

Esta seção contém informações sobre como diagnosticar e corrigir problemas de desenvolvimento com seu aplicativo Managed Service for Apache Flink.

Tópicos

Melhores práticas de reversão do sistema

Com recursos automáticos de reversão do sistema e visibilidade das operações no Amazon Managed Service para Apache Flink, você pode identificar e resolver problemas com seus aplicativos.

Reversões do sistema

Se a operação de atualização ou escalabilidade do seu aplicativo falhar devido a um erro do cliente, como um bug de código ou problema de permissão, o Amazon Managed Service para Apache Flink tentará automaticamente reverter para a versão em execução anterior se você tiver optado por essa funcionalidade. Para obter mais informações, consulte Habilite reversões do sistema para seu aplicativo Managed Service for Apache Flink. Se essa reversão automática falhar ou você não tiver optado por participar ou não, seu aplicativo será colocado no estado. `READY` Para atualizar seu aplicativo, conclua as seguintes etapas:

Reversão manual

Se o aplicativo não estiver progredindo e estiver em um estado transitório por muito tempo, ou se a transição do aplicativo for bem-sucedida`Running`, mas você observar problemas posteriores, como erros de processamento em um aplicativo Flink atualizado com êxito, você poderá revertê-lo manualmente usando o. `RollbackApplication` API

1. Chamada `RollbackApplication` - isso reverterá para a versão anterior em execução e restaurará o estado anterior.
2. Monitore a operação de reversão usando o. `DescribeApplicationOperation` API
3. Se a reversão falhar, use as etapas anteriores de reversão do sistema.

Visibilidade das operações

`ListApplicationOperations`APIMostra o histórico de todas as operações do cliente e do sistema em seu aplicativo.

1. Obtenha o operationIdda operação com falha na lista.
2. Ligue `DescribeApplicationOperation` e verifique o status statusDescriptione.
3. Se uma operação falhar, a descrição aponta para um possível erro a ser investigado.

Bugs comuns de código de erro: use os recursos de reversão para reverter para a última versão em funcionamento. Resolva os bugs e repita a atualização.

Problemas de permissão: use o `DescribeApplicationOperation` para ver as permissões necessárias. Atualize as permissões do aplicativo e tente novamente.

Amazon Managed Service para problemas com o serviço Apache Flink: verifique AWS Health Dashboard ou abra um caso de suporte.

Melhores práticas de configuração do Hudi

Para executar conectores Hudi no Managed Service for Apache Flink, recomendamos as seguintes alterações de configuração.

Desativar `hoodie.embed.timeline.server`

O conector Hudi no Flink configura um servidor de linha do tempo (TM) incorporado no gerenciador de tarefas do Flink (JM) para armazenar metadados em cache e melhorar o desempenho quando o paralelismo de tarefas é alto. Recomendamos que você desabilite esse servidor incorporado no Managed Service for Apache Flink porque desabilitamos a comunicação não-Flink entre JM e TM.

Se esse servidor estiver habilitado, o Hudi Writes primeiro tentará se conectar ao servidor incorporado no JM e, em seguida, voltará a ler os metadados do Amazon S3. Isso significa que o Hudi incorre em um tempo limite de conexão que atrasa as gravações do Hudi e causa um impacto no desempenho do Managed Service for Apache Flink.

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 eles. 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 Flame Graphs para seu aplicativo, crie um caso para solicitar que ele seja desativado para seu aplicativoARN. Para obter mais informações, consulte o AWS Support Center.

Problema do provedor de credenciais com o EFO conector 1.15.2

Há um problema conhecido com as versões do conector Kinesis EFO Data Streams até 1.15.2 em que o não `FlinkKinesisConsumer` está respeitando a configuração. `Credential Provider` As configurações válidas são 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 ao Kinesis EFO usando o conector.

Para resolver esse erro, use a versão 1.15.3 ou superior do EFO conector.

Aplicativos com conectores Kinesis não compatíveis

O Managed Service for Apache Flink for Apache Flink versão 1.15 ou posterior rejeitará automaticamente a inicialização ou atualização dos aplicativos se eles estiverem usando versões incompatíveis do Kinesis Connector (pré-versão 1.15.2) agrupadas no aplicativo ou nos arquivos (). JARs 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 de 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 “Usando o conector Apache Flink Kinesis Streams com versões anteriores do Apache Flink”.

Escolha inválida: “kinesisanalyticsv2"

Para usar a v2 do Managed Service para Apache FlinkAPI, você precisa da versão mais recente do AWS Command Line Interface ().AWS CLI

Para obter informações sobre como atualizar o AWS CLI, consulte Instalando o AWS Command Line Interface no Guia do AWS Command Line Interface Usuário.

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 pelos sistemas de arquivos. 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.

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 MPU ciclo de vida do S3 Bucket 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 a versão mais recente do Managed Service for Apache Flink 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 STOPPING ou UPDATINGSCALING, 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 o `FlinkKafkaConsumer` com instantâneos habilitados, existe a possibilidade de que, quando a tarefa do Flink processe uma parada com uma API solicitação de ponto de salvamento, ela falhe com um erro `FlinkKafkaConsumer` de tempo de execução relatando a. `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 SQL API Flink/Tabel/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 que faz AWS SDK com que certos erros não sejam exibidos 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 a AWS SDK 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 em 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 a AWS SDK, no jarro do conector. Para atualizar a AWS SDK versão, use o procedimento a seguir para substituir essas classes sombreadas:

Maven

1. Adicione o conector Kinesis e os AWS SDK módulos necessários como dependências do projeto.
2. Configure `maven-shade-plugin`:

    a. Adicione um filtro para excluir AWS SDK classes sombreadas ao copiar o conteúdo do jar do conector Kinesis.

    b. Adicione uma regra de realocação para mover AWS SDK classes atualizadas para o pacote, o que é 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 AWS SDK módulos necessários como dependências do projeto.
2. Ajuste a shadowJar configuração:

   a. Exclua AWS SDK classes sombreadas ao copiar o conteúdo do jar do conector Kinesis.

   b. Realoque AWS SDK as classes 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("software/amazon/kinesis/shaded/software/amazon/awssdk/**/*")
    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 AWS SDK versão, ela deve ser aplicada na configuração de compilação do projeto. SDK

Maven

Adicione a AWS SDK lista de materiais (BOM) à seção de gerenciamento de dependências do `pom.xml` arquivo para impor a SDK versão 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 a dependência da plataforma na AWS SDK lista de materiais (BOM) para impor a SDK versão 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 do conector, especificando a versão necessária 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 do conector, especificando a versão necessária 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 do conector, especificando a versão necessária 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 do conector, especificando a versão necessária AWS SDK. Para acelerar a compilação, use `-DskipTests` para ignorar a execução do teste e `-Dfast` para ignorar verificações adicionais do código-fonte:

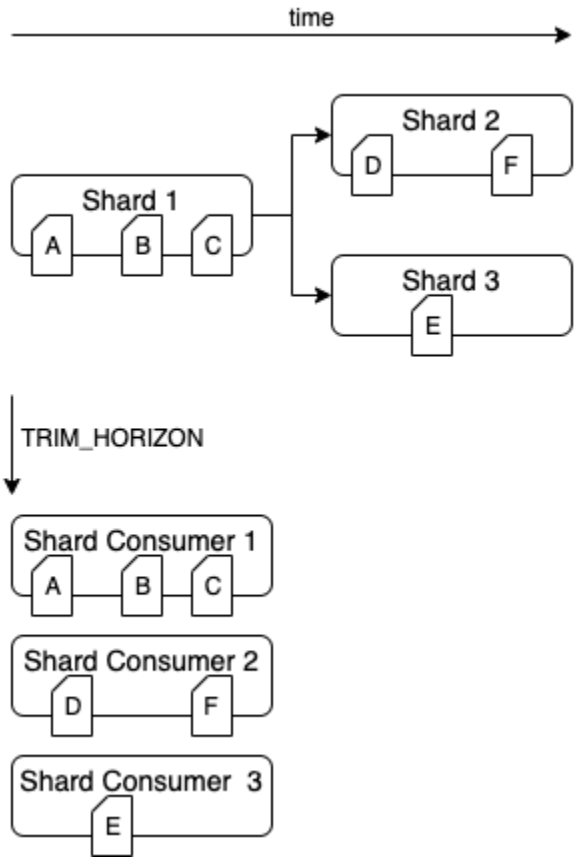
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -
Daws.sdk.version=2.20.144
```

5. O jar resultante estará disponível em:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

Streams de dados do Amazon Kinesis: processamento da fonte 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 execução

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 CloudWatch alarmes. Usando CloudWatch alarmes, você pode definir limites para CloudWatch métricas que indicam condições de erro ou gargalo em seu aplicativo. Para obter informações sobre CloudWatch os alarmes recomendados, consulteUse CloudWatch alarmes com o Amazon Managed Service para Apache Flink.

Problemas com o aplicativo

Esta seção contém soluções para condições de erro que você pode encontrar com seu aplicativo Managed Service for Apache Flink.

Tópicos

- Não é possível acessar recursos em um VPC
- Os dados são perdidos ao gravar em um bucket do Amazon S3
- O aplicativo está no RUNNING status, mas não está processando dados
- Erro de captura instantânea, atualização do aplicativo ou parada do aplicativo: InvalidApplicationConfigurationException
- java.nio.file. NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

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 StopApplicationação com o `Force` parâmetro 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 snapshot bem-sucedido. Se o aplicativo não tiver snapshots, o Managed Service for Apache Flink rejeitará a solicitação de reversão. Para obter mais informações sobre como reverter um aplicativo, consulte RollbackApplicationação.

> **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 JAR arquivo do aplicativo tem menos de 512 MB. JARarquivos 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 AllowNonRestoredStateparâmetro do aplicativo FlinkRunConfigurationpara`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 seu trabalho não começar dentro desse tempo limite, você verá um CloudWatch registro 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 JobManagerregistros do Flink, bem como o código do aplicativo, para ver se esse atraso no `main` método 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 Gerencie backups de aplicativos usando instantâneos.
- 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 um VPC

Se seu aplicativo usa uma VPC execução na AmazonVPC, faça o seguinte para verificar se seu aplicativo tem acesso aos recursos:

- Verifique se há o seguinte erro em seus CloudWatch registros. Esse erro indica que seu aplicativo não pode acessar recursos em seuVPC:

  ```
  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 como configurar e analisar CloudWatch registros, consulteRegistro e monitoramento no Amazon Managed Service para Apache Flink.

Os 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 Firehose. Para obter mais informações sobre o uso do Firehose com o Managed Service para Apache Flink, consulte. Pia Firehose

O aplicativo está no RUNNING status, mas não está processando dados

Você pode verificar o status do seu aplicativo usando as ações `ListApplications` ou `DescribeApplication`. Se seu aplicativo inserir o `RUNNING` status, mas não estiver gravando dados no coletor, você poderá solucionar o problema adicionando um stream de CloudWatch log da Amazon ao seu aplicativo. Para obter mais informações, consulte Trabalhe com opções de CloudWatch registro de aplicativos. O fluxo de logs contém mensagens que podem ajudar a solucionar problemas do aplicativo.

Erro de captura instantânea, 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 `SnapshotsEnabledUpdate` parâmetro da UpdateApplicationação.
- 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 SSL armazenamento confiável 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 registro do seu aplicativo para essas mudanças de status usando a seguinte consulta do CloudWatch Logs Insights:Analise 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 ter mais informações, consulte the section called “Analise erros: falhas relacionadas à tarefa do aplicativo”.

- Os streams de dados do Kinesis não estão provisionados adequadamente: se uma fonte ou coletor do seu aplicativo for um stream de dados do Kinesis, verifique se há erros ou métricas do stream. `ReadProvisionedThroughputExceeded` `WriteProvisionedThroughputExceeded`

  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 AWS serviços ou fontes ou destinos externos) estão bem provisionados, não estão sofrendo limitação de leitura ou gravação 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 registro 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 registro do aplicativo se seus 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 CloudWatch métricas quanto a picos incomuns no uso da CPU ou da memória.
  - Verifique se há problemas de throughput em seu aplicativo. Para ter mais informações, consulte Solucionar 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 Não encontrado: esse erro ocorre se seu aplicativo usa o Apache Beam, mas não tem as dependências ou versões de dependências 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. Use CloudFormation com o serviço gerenciado para 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.

A taxa de transferência é muito lenta

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 MSK cluster da Amazon, as métricas de atraso do consumidor do cluster aumentam continuamente. Para obter mais informações, consulte Consumer-Lag Monitoring no Amazon Developer Guide. 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 Solucionar 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 Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.
- 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 MSKConfigurações personalizadas.

Para obter as etapas de solução de problemas relacionados ao throughput baixo ou ao aumento do atraso do consumidor na origem do aplicativo, consulte Solucionar problemas de desempenho.

Crescimento estadual ilimitado

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 ampliando sua BeamApplicationProperties com os principais experimentos e valoresuse_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 AggregateFunctionssejam usados, o que permite 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 estado time-to-live 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 `GetRecords` chamada (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.

Você pode confirmar isso verificando a CloudWatch `ReadProvisionedThroughputExceeded` métrica e vendo períodos prolongados ou sustentados em que essa métrica é maior que zero.

Você também pode ver isso nos CloudWatch registros do seu aplicativo Amazon Managed Service for Apache Flink observando erros contínuos. `LimitExceededException`

Resolução: Você pode fazer uma das duas coisas para resolver esse cenário:

- Diminua o limite padrão para o número de registros buscados por chamada `GetRecords`

- Habilite leituras adaptáveis em seu aplicativo Amazon 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.
- Reconhecer — Geralmente é um estágio curto e é simplesmente a subtarefa de enviar uma confirmação para o JobManager e também realizar 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:

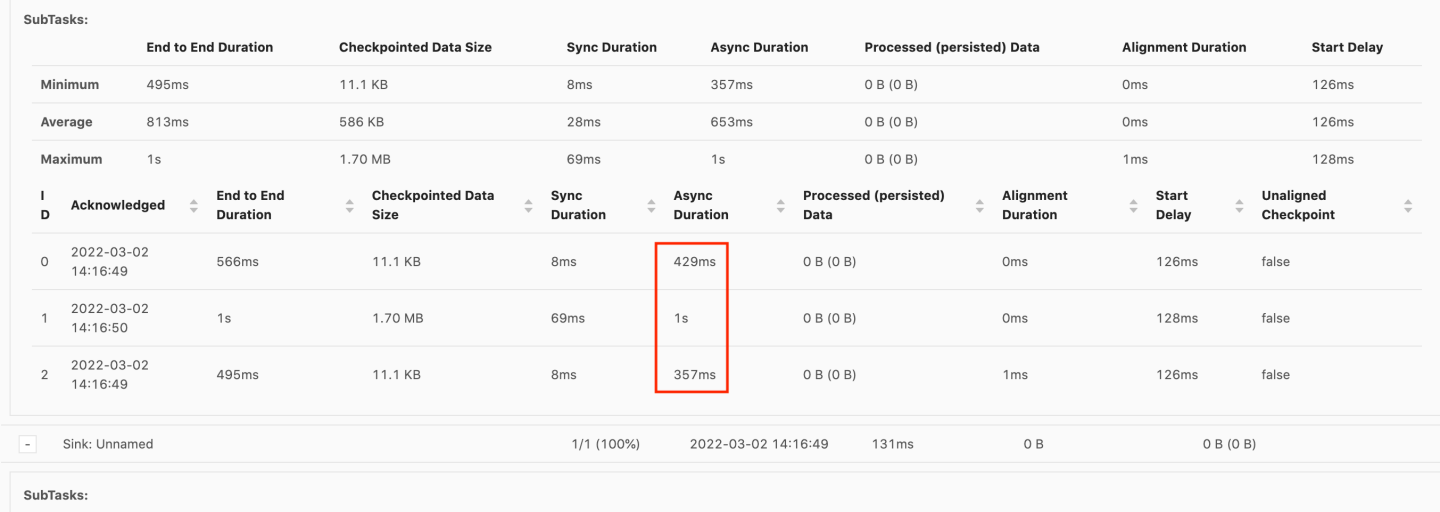
A duração do timestamp do acionador até a confirmação mais recente (ou n/a, se nenhuma confirmação foi recebida ainda). Essa duração de ponta a ponta para um ponto de verificação completo é determinada pela última subtarefa que reconhece o ponto de verificação. Esse tempo geralmente é maior do que as 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()` são chamadas classes que implementam a snapshotState interface; isso pode ser um código de usuário, portanto, thread-dumps 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:

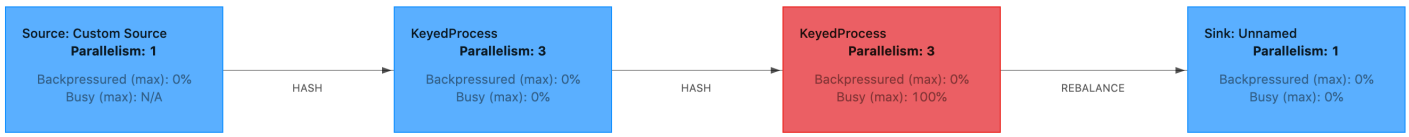
	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay
Minimum	495ms	11.1 KB	8ms	357ms	0 B (0 B)	0ms	126ms
Average	813ms	586 KB	28ms	653ms	0 B (0 B)	0ms	126ms
Maximum	1s	1.70 MB	69ms	1s	0 B (0 B)	1ms	128ms

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 JobGraph em que o segundo KeyedProcess operador está ocupado.



Você pode investigar o que está demorando tanto usando 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 JVM encadeamento, que é um estado de execução de todos os encadeamentos dentro do 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 tópico KeyedProcess (1/3) #0
- Prioridade do thread prio=5
- Uma ID de thread exclusiva Id=1423
- Estado do tópico RUNNABLE

O nome de um thread geralmente fornece informações sobre o propósito geral do thread. Os encadeamentos do operador podem ser identificados pelo nome, pois os encadeamentos do operador têm o mesmo nome do operador, bem como uma indicação da subtarefa à qual estão

relacionados, por exemplo, o encadeamento KeyedProcess (1/3) #0 é do KeyedProcessoperador e da 1ª (de 3) subtarefa.

Os threads podem estar em um dos seguintes estados:

- NEW— O tópico foi criado, mas ainda não foi processado
- RUNNABLE— O tópico é executado no CPU
- BLOCKED— O tópico está esperando que outro encadeamento libere seu cadeado
- WAITING— O tópico está aguardando usando um `park()` método `wait()join()`, ou
- TIMED_ WAITING — O tópico está aguardando usando o método sleep, wait, join ou park, mas com um tempo máximo de espera.

> **Note**
>
> No Flink 1.13, a profundidade máxima de um único rastreamento de pilha no despejo de thread é limitada a 8.

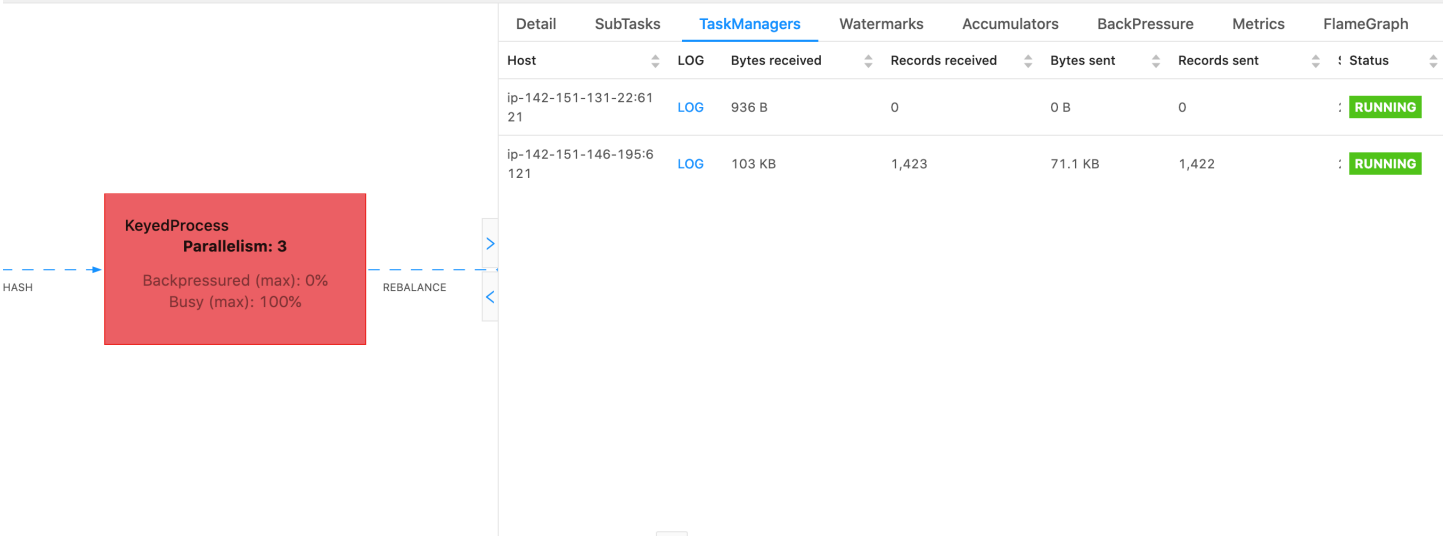
> **Note**
>
> Os despejos de thread devem ser o último recurso para depurar problemas de desempenho em um aplicativo Flink, pois podem ser difíceis de ler e exigem que várias amostras sejam coletadas e analisadas manualmente. Se possível, é preferível usar Flame graphs.

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 tópicos da TaskManagersguia 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 KeyedProcessoperador 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
    ...
```

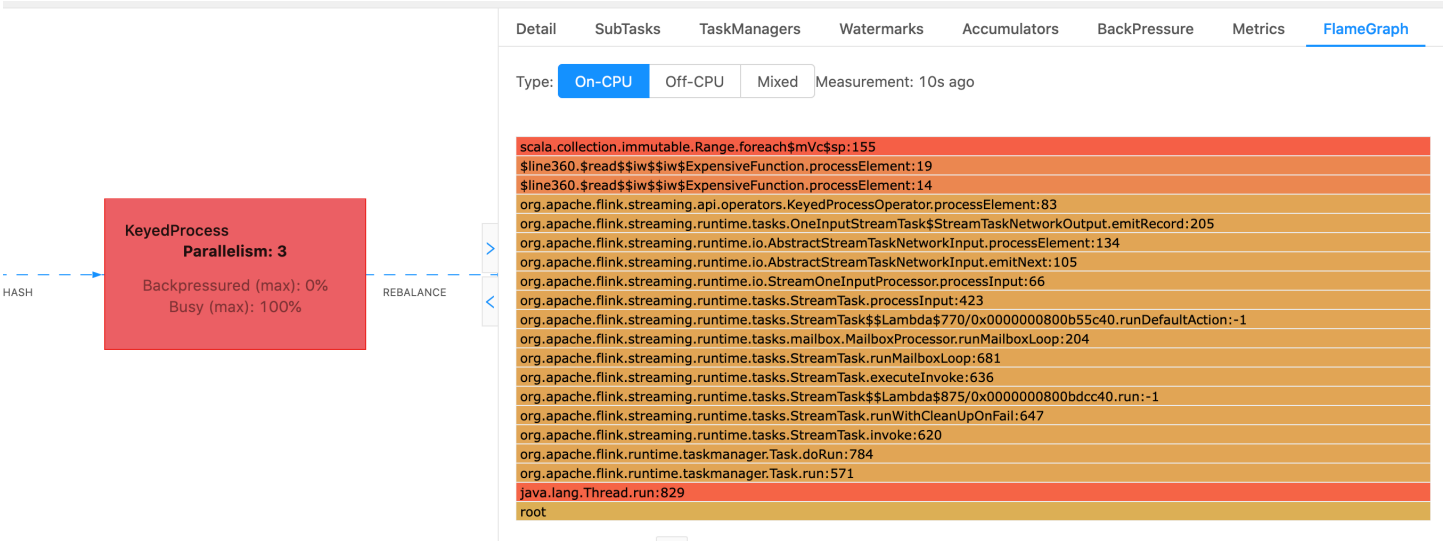
Isso pode se tornar confuso se houver vários operadores com o mesmo nome, mas podemos nomear operadores para contornar isso. Por exemplo:

```
....
.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 gráfico de chama de um operador pode ser acessado por meio da interface do usuário da Web selecionando um operador e, em seguida, escolhendo a FlameGraphguia. Depois que amostras suficientes forem coletadas, o Flame graph será exibido. A seguir está a FlameGraph versão ProcessFunction que estava demorando muito para chegar ao posto de controle.



Este é um gráfico de chamas muito simples e mostra que todo o CPU tempo está sendo gasto em uma visão geral `processElement` do ExpensiveFunction operador. 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 Tipos de dados e serialização 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 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 KPU provisionamento insuficiente. Para obter informações sobre como aumentar o provisionamento de aplicativos, consulte Implemente o escalonamento de aplicativos no Managed Service para Apache Flink.
- 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 shutdownSourcesAfterIdleMs0 ms, os pontos de verificação podem falhar ao serem acionados porque as tarefas estão no estado FINISHED "". Esta seção descreve os sintomas e a resolução dessa condição.

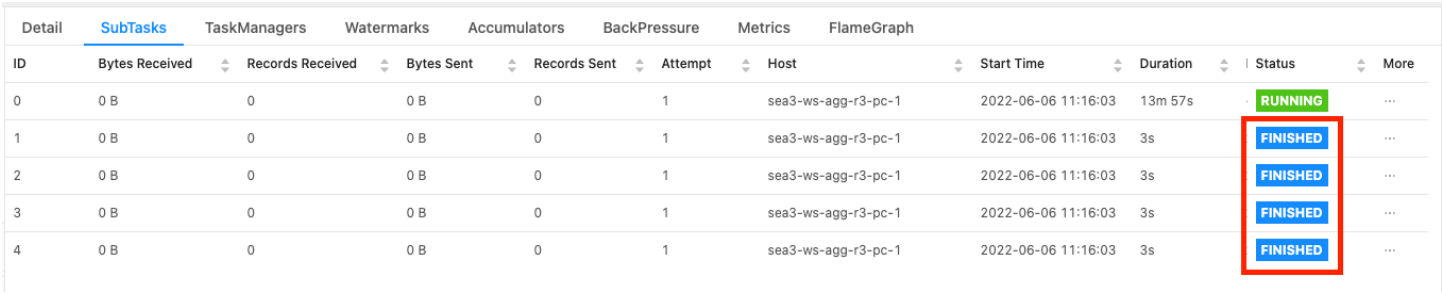
Sintomas

Acesse os CloudWatch registros 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 "FINISHED" e o ponto de verificação não é mais possível.



Causa

shutdownSourcesAfterIdleMs é uma variável de configuração do Beam que desliga fontes que ficaram ociosas 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 FINISHED "" é quando shutdownSourcesAfter IdleMs está definido como 0 ms, o que significa que as tarefas que estão ociosas serão encerradas imediatamente.

Solução

Para evitar que as tarefas entrem no estado FINISHED "" imediatamente, shutdownSourcesAfter IdleMs defina como Longo. 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 shutdpwnSourcesAfteridle Ms da seguinte forma:

- Opção 2: Se a configuração do feixe estiver definida em seu JAR arquivo, você poderá definir da shutdownSourcesAfter IdleMs 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 CPU recursos 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 escalar para trabalho limitado a CPU /memória ou aumentar `ParallelismPerKPU` para trabalho limitado a 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 a produtividade 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.

Integre-se 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 Support Centro.

Histórico de documentos do Amazon Managed Service para 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 Crie	22 de novembro de 2022

Alteração	Descrição	Data
	um serviço gerenciado para o aplicativo Apache Flink.	
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 Use Python com serviço gerenciado para Apache Flink.	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 Crie um serviço gerenciado para o aplicativo Apache Flink.	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 Use o painel do Apache Flink com serviço gerenciado para o 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 Use CloudFormation com o serviço gerenciado para 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 ???.	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 ???.	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 ???.	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 ???.	12 de maio de 2020
Exemplo: Leia a partir de um fluxo de dados do Kinesis em outra conta.	Saiba como acessar um stream do Kinesis em uma AWS conta diferente em seu aplicativo Managed Service for Apache Flink. Para ter mais informações, consulte Entre contas.	30 de março de 2020

Alteração	Descrição	Data
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 Crie um serviço gerenciado para o aplicativo Apache Flink.	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 Configure o Managed Service para o Apache Flink para acessar recursos em 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 ???.	14 de outubro de 2019
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 ???.	26 de junho de 2019

Alteração	Descrição	Data
Propriedades de runtime do aplicativo do Managed Service for Apache Flink	Trabalhe com as propriedades de runtime no Managed Service for Apache Flink. Para ter mais informações, consulte Use propriedades de tempo de execução no Managed Service para Apache Flink.	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 Adicionar tags ao Managed Service para aplicativos Apache Flink.	8 de maio de 2019
Registrando o serviço gerenciado para chamadas da API Apache Flink com AWS CloudTrail	O Managed Service for Apache Flink é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Managed Service for Apache Flink. Para ter mais informações, consulte ???.	22 de março de 2019

Alteração	Descrição	Data
Criar um aplicativo (Firehose Sink)	Faça um exercício para criar um serviço gerenciado para o Apache Flink com um stream de dados do Amazon Kinesis como fonte e um stream do Amazon Data Firehose como coletor. Para ter mais informações, consulte Pia 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

Código de exemplo do Managed Service for Apache Flink API

Este tópico contém exemplos de blocos de solicitação para ações no Managed Service for Apache Flink.

Para usar o 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 o AWS CLI, consulte Gerar parâmetros JSON do esqueleto da CLI e da entrada da CLI no Guia do usuário.AWS Command Line Interface

Tópicos

- DescribeApplication
- DescribeApplicationSnapshot
- DiscoverInputSchema
- ListApplications
- ListApplicationSnapshots
- StartApplication
- StopApplication
- UpdateApplication

AddApplicationCloudWatchLoggingOption

O exemplo de código de solicitação a seguir para a AddApplicationCloudWatchLoggingOptionação adiciona uma opção de CloudWatch registro da Amazon a um aplicativo 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 AddApplicationInputação adiciona uma entrada de aplicativo a um aplicativo 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 AddApplicationInputProcessingConfigurationação adiciona uma configuração de processamento de entrada do aplicativo a um serviço gerenciado para o aplicativo 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 AddApplicationOutputação adiciona um stream de dados do Kinesis como saída do aplicativo para um serviço gerenciado para o aplicativo 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 AddApplicationReferenceDataSourceação adiciona uma fonte de dados de referência do aplicativo CSV a um aplicativo 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 a seguir de código de solicitação para a AddApplicationVpcConfigurationação adiciona uma configuração de VPC a um aplicativo existente:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

O exemplo de código de solicitação a seguir para a CreateApplicationação cria um serviço gerenciado para o aplicativo 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 CreateApplicationSnapshotação cria um instantâneo do estado do aplicativo:

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

DeleteApplication

O exemplo de código de solicitação a seguir para a DeleteApplicationação exclui um serviço gerenciado para o aplicativo Apache Flink:

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

O exemplo de código de solicitação a seguir para a DeleteApplicationCloudWatchLoggingOptionação exclui uma opção de CloudWatch registro da Amazon de um aplicativo Managed Service for Apache Flink:

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

```
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

O exemplo de código de solicitação a seguir para a DeleteApplicationInputProcessingConfigurationação remove uma configuração de processamento de entrada de um aplicativo Managed Service for Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

O exemplo de código de solicitação a seguir para a DeleteApplicationOutputação remove a saída de um aplicativo do Managed Service for Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

O exemplo de código de solicitação a seguir para a DeleteApplicationReferenceDataSourceação remove uma fonte de dados de referência do aplicativo Managed Service for Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

O exemplo de código de solicitação a seguir para a DeleteApplicationSnapshotação exclui um instantâneo do estado do aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

O exemplo a seguir de código de solicitação para a DeleteApplicationVpcConfigurationação remove 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 DescribeApplicationação retorna detalhes sobre um serviço gerenciado para o aplicativo Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

O exemplo de código de solicitação a seguir para a DescribeApplicationSnapshotação retorna detalhes sobre um instantâneo do estado do aplicativo:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
```

```
}
```

DiscoverInputSchema

O exemplo de código de solicitação a seguir para a DiscoverInputSchemaação gera um esquema de uma fonte de streaming:

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "NOW"
   },
   "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string"
   },
   "ServiceExecutionRole": "string"
}
```

O exemplo de código de solicitação a seguir para a DiscoverInputSchemaação 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 ListApplicationsação retorna uma lista de serviços gerenciados para aplicativos Apache Flink em sua conta:

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

ListApplicationSnapshots

O exemplo de código de solicitação a seguir para a ListApplicationSnapshotsação retorna uma lista de instantâneos do estado do aplicativo:

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

O exemplo de código de solicitação a seguir para a StartApplicationação 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 a seguir, o código de solicitação para a StopApplication ação API_ interrompe um serviço gerenciado para o aplicativo Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

O exemplo de código de solicitação a seguir para a UpdateApplicationação 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.

Esse conteúdo foi movido para as versões Release. Consulte Versões de liberação.