Seminário 3

Engenharia de Software - 8º Período

Aluno:
Cláudio da Silva Leite
Eric Yuji Ikeda
João Vitor Cunha

Professor:
Mauricio

Disciplina
Arquitetura de Sistemas IoT e Cloud Computing

Data:

Introdução

O objetivo deste projeto, demonstra a integração entre containers Docker, Apache Kafka e Cloud Functions. A proposta é criar um sistema composto por dois containers, um Publisher e um Subscriber, que se comunicam através de um serviço de streaming de mensagens, o Kafka, enquanto uma Cloud Function atua como um orquestrador, acionando o Publisher ou interagindo com o Kafka para desencadear eventos com base em triggers, como requisições HTTP ou mudanças de estado. O projeto busca validar a viabilidade técnica dessa arquitetura, verificando a comunicação eficiente entre esses componentes em um ambiente containerizado.

Os componentes principais utilizados no projeto são o Docker, o Apache Kafka e as Cloud Functions. O Docker é uma plataforma de conteinerização que facilita a criação de ambientes isolados e padronizados para o desenvolvimento e execução de aplicações. O Kafka é um serviço de streaming de mensagens que permite o processamento de grandes volumes de dados em tempo real, conectando produtores (Publisher) e consumidores (Subscriber) através de tópicos. Já as Cloud Functions são funções serverless que executam código em resposta a eventos, oferecendo escalabilidade automática e reduzindo a necessidade de gerenciamento de infraestrutura. Essas tecnologias, combinadas, possibilitam uma arquitetura robusta para o desenvolvimento de aplicações distribuídas e escaláveis.

Requisitos de Ambiente

Para implementar no projeto foram necessárias algumas ferramentas essenciais. O Docker é utilizado para criar e gerenciar os containers onde o Publisher e o Subscriber serão executados de forma isolada. O Docker Compose é empregado para orquestrar os serviços, permitindo que o Kafka, o Zookeeper, e os containers Publisher e Subscriber sejam iniciados e configurados de maneira automatizada. A linguagem de programação escolhida, como Python, facilita o desenvolvimento das lógicas de publicação e consumo de mensagens, utilizando bibliotecas como kafka-python para interagir com o Apache Kafka. Além disso, o Docker permite o empacotamento dessas aplicações em ambientes replicáveis e portáveis.

Além das ferramentas locais, é necessário ter acesso a um serviço de Cloud Functions, como o Google Cloud Functions, AWS Lambda, ou Azure Functions. Esses serviços serverless permitem a execução de código em resposta a eventos, como requisições HTTP ou mudanças de estado. A Cloud Function será configurada para acionar o Publisher ou interagir diretamente com o Kafka, orquestrando o fluxo de mensagens entre os containers. O uso de uma função na nuvem elimina a necessidade de gerenciamento de infraestrutura, permitindo que o código seja executado automaticamente conforme os eventos ocorrem, garantindo escalabilidade e simplicidade na arquitetura.

Arquitetura da Solução

A arquitetura da solução envolve quatro componentes principais: o Publisher, o Kafka, o Subscriber e a Cloud Function. O Publisher é responsável por enviar mensagens para um tópico no Kafka, enquanto o Subscriber consome essas mensagens para processá-las. Ambos são executados em containers Docker, garantindo ambientes isolados e portáteis. O Kafka atua como intermediário, gerenciando o fluxo de mensagens entre o Publisher e o Subscriber por meio de tópicos. A Cloud Function serve como um orquestrador, acionando o Publisher ou interagindo com o Kafka sempre que um evento, como uma requisição HTTP, é recebido. Esse fluxo permite a comunicação fluida e escalável entre os componentes, utilizando a Cloud Function para garantir automação e flexibilidade no envio de mensagens.

O fluxo de dados segue a seguinte sequência: a Cloud Function é acionada por um evento, como uma requisição HTTP ou mudança de estado. Em resposta, ela dispara uma lógica que aciona o Publisher para publicar uma mensagem em um tópico no Kafka. O Kafka recebe e armazena essa mensagem no tópico definido. Em seguida, o Subscriber consome a mensagem, processando-a de acordo com a lógica implementada. Esse fluxo garante que as mensagens fluem de maneira assíncrona e distribuída, aproveitando as capacidades do Kafka para lidar com grandes volumes de dados e das Cloud Functions para oferecer uma orquestração automática e escalável.

Objetivo

O objetivo deste projeto é demonstrar a integração entre containers Docker, Apache Kafka e uma Cloud Function para gerenciar o fluxo de mensagens entre um Publisher e um Subscriber.

A proposta é que o Publisher envie mensagens para um tópico no Kafka, que serão consumidas pelo Subscriber, enquanto a Cloud Function pode acionar o processo de publicação de forma serverless, conforme determinado por eventos ou triggers específicos.

Estrutura da Solução

O projeto é composta por três componentes principais:

Containers Docker:

Publisher: Um container que publica mensagens em um tópico Kafka.

Subscriber: Um container que consome mensagens desse mesmo tópico.

Apache Kafka: Um serviço de streaming de dados responsável por gerenciar a troca de mensagens entre os containers.

Cloud Function: Um serviço que aciona o Publisher ou interage diretamente com o Kafka, permitindo a execução de ações serverless baseadas em eventos como requisições HTTP.

A comunicação entre esses componentes é baseada no uso de Docker para isolar os ambientes, Kafka para garantir a entrega de mensagens, e Cloud Functions para executar ações serverless. A PoC busca validar o correto funcionamento desse fluxo e a integração eficiente entre eles.

Requisitos de Ambiente

Para implementar nesse projeto, são necessárias as seguintes ferramentas:

Docker e Docker Compose: Para configurar e gerenciar os containers.

Linguagem de Programação: Python foi escolhido por sua simplicidade e ampla biblioteca para integração com Kafka.

Plataforma de Cloud Functions: Pode-se utilizar Google Cloud Functions, AWS Lambda ou Azure Functions, dependendo da escolha da infraestrutura de nuvem.

Arquitetura da Solução

A arquitetura envolve três principais etapas:

Publicação de Mensagens pelo Publisher: O Publisher é responsável por enviar mensagens para um tópico Kafka. Ele é desenvolvido utilizando Python com a biblioteca kafka-python.

Consumo de Mensagens pelo Subscriber: O Subscriber consome e processa as mensagens publicadas no Kafka, também utilizando Python.

Integração com Cloud Functions: Uma Cloud Function é configurada para acionar o Publisher ou realizar uma interação direta com o Kafka, respondendo a eventos como mudanças de estado ou triggers HTTP.

O fluxo de dados começa quando uma requisição HTTP é enviada à Cloud Function, que aciona o Publisher para publicar uma mensagem no Kafka. O Kafka armazena a mensagem no tópico, e o Subscriber consome essa mensagem e processa-a conforme necessário.

Passo a Passo de Configuração e Implementação

Configuração do Kafka com Docker Compose: O Kafka é configurado dentro de um ambiente Docker usando docker-compose.yml, que também inclui o Zookeeper, necessário para a coordenação do Kafka.

Exemplo de configuração básica do Docker Compose:

```
version: '3'
services:
  zookeeper:
    image: wurstmeister/zookeeper:3.4.6
    ports:
      - "2181:2181"
  kafka:
    image: wurstmeister/kafka:2.12-2.2.1
    ports:
      - "9092:9092"
    environment:
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
```

Desenvolvimento do Publisher e do Subscriber:

Publisher: O código do Publisher foi desenvolvido para enviar mensagens ao Kafka. Ele utiliza a biblioteca kafka-python para se comunicar com o Kafka, enviando dados em JSON para um tópico configurado.

Subscriber: O Subscriber consome as mensagens do mesmo tópico e imprime ou processa os dados.

Exemplo de código do Publisher:

```
from kafka import KafkaProducer
import json

producer = KafkaProducer(bootstrap_servers='localhost:9092',
                         value_serializer=lambda v: json.dumps(v).encode('utf-8'))

def publish_message(message):
    producer.send('test-topic', value=message)
    producer.flush()
```

```
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer('test-topic',
                         bootstrap_servers='localhost:9092',
                         auto_offset_reset='earliest',
                         enable_auto_commit=True,
                         value_deserializer=lambda x: json.loads(x.decode('utf-8')))

for message in consumer:
    print(f"Received message: {message.value}")
```

Exemplo de código do Subscriber:

```
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer('test-topic',
```

```
    bootstrap_servers='localhost:9092',
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    value_deserializer=lambda x: json.loads(x.decode('utf-8')))

for message in consumer:
    print(f"Received message: {message.value}")
```

Desenvolvimento da Cloud Function: A Cloud Function é responsável por acionar o Publisher ou interagir diretamente com o Kafka. Ela pode ser acionada por uma requisição HTTP contendo a mensagem a ser publicada no Kafka.

Exemplo de código para a Cloud Function (em Python):

```
import requests

def publish_message(request):
    message = request.args.get('message')
    if not message:
        return 'No message provided', 400

    kafka_url = 'http://localhost:9092'
    # Aciona o publisher via HTTP ou executa lógica de publicação no Kafka
    requests.post(f"{kafka_url}/publish", json={'message': message})

    return 'Message published successfully!', 200
```

Deploy e Testes:

- O ambiente Docker pode ser iniciado usando o comando docker-compose up -d.
- A Cloud Function pode ser implementada seguindo as instruções da plataforma escolhida (Google Cloud Functions, AWS Lambda, etc.).
- Para testar o sistema, faça uma requisição HTTP para a Cloud Function com uma mensagem e verifique se o Subscriber consumiu essa mensagem.

Resultados Esperados

As mensagens publicadas pelo Publisher devem ser enviadas corretamente para o Kafka e consumidas pelo Subscriber.

A Cloud Function deve ser capaz de acionar o processo de publicação no Kafka ou interagir diretamente com ele, dependendo do evento configurado.

O ambiente do projeto deve demonstrar a comunicação fluida entre os componentes, validando a integração entre containers Docker, Kafka e Cloud Functions. Os próximos passos podem incluir otimizações para ambientes de produção, como escalabilidade, segurança e monitoração.

Bibliografia

Data: 09/09/2024 às 19:00 h.
SQLStreamify- Middleware baseado em microsserviços fornecendo transações de leitura de fluxo de dados em SGBDs

https://repositorio.unesp.br/server/api/core/bitstreams/91d360a2-9ab6-4963-84b5-c1cda581c92d/content

Data: 14/09/2024 às 08:00 h
Desenvolvimento da Infraestrutura de um Espaço Inteligente baseado em Visão Computacional e IoT
https://eletrica.ufes.br/sites/engenhariaeletrica.ufes.br/files/field/anexo/felippe_m_queiroz.pdf

Data: 16/09/2024 às 19:00 h.
Noções básicas sobre o Cloud Tasks
https://cloud.google.com/tasks/docs/dual-overview?hl=pt-br