Seminário 4

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 Computig

Data:
25/09/2024

Padrões de Mensageria em Arquiteturas Distribuídas na Nuvem

As arquiteturas modernas de sistemas distribuídos enfrentam desafios como alta disponibilidade, escalabilidade e consistência. Para lidar com essas questões, o uso de padrões de mensageria é essencial. Eles permitem a comunicação entre componentes de maneira assíncrona, desacoplada e confiável, facilitando a integração de serviços e sistemas na nuvem.

1. Tipos de Padrões de Mensageria

Fila de Mensagens (Message Queue): Um padrão onde as mensagens são enviadas para uma fila, permitindo a comunicação assíncrona entre produtor e consumidor. Exemplos de uso incluem o gerenciamento de cargas de trabalho ou processamento de pedidos. Em Azure, esse conceito é implementado no Azure Queue Storage e no Azure Service Bus.

Publicar/Assinar (Pub/Sub): Um mecanismo onde produtores enviam mensagens a vários consumidores sem saber quem eles são, através de um tópico. Isso melhora o desacoplamento. No Azure, esse padrão é suportado pelo Azure Event Grid e pelo Azure Service Bus Topics.

Barramento de Serviço (Service Bus): Facilita a comunicação entre aplicativos e serviços por meio de uma infraestrutura robusta que garante entrega confiável de mensagens. No Azure Service Bus, há recursos como filas, tópicos e assinaturas.

Eventos de Integração (Event Integration): Usado para lidar com eventos entre sistemas. Esses eventos são disparados quando algo acontece, como a criação de uma nova ordem em um sistema ERP, o que aciona outros serviços ou workflows. Azure Event Hubs e Event Grid são soluções para eventos em escala.

2. Benefícios do Uso de Padrões de Mensageria

Desacoplamento: Ao utilizar filas e mecanismos de pub/sub, os componentes do sistema não dependem diretamente uns dos outros. Isso facilita a manutenção e a escalabilidade.

Escalabilidade: Mensageria permite escalar serviços independentemente. Por exemplo, se o consumidor de mensagens estiver sobrecarregado, ele pode processar as mensagens em seu próprio ritmo.

Resiliência: As mensagens são armazenadas em filas até que sejam processadas, garantindo que nenhuma informação se perca mesmo em caso de falha de algum componente.

3. Cenários de Uso

Os padrões de mensageria são amplamente aplicados em vários cenários:

Processamento de pedidos em sistemas de e-commerce, onde os pedidos podem ser enviados para uma fila e processados assíncronamente.

Integração entre sistemas legados e modernos, onde eventos de um sistema antigo são capturados e propagados para novas aplicações.

Microserviços, que usam mensageria para comunicação entre componentes sem dependências diretas, facilitando a atualização e o desenvolvimento contínuo.

Conclusão

Os padrões de mensageria são fundamentais para a construção de arquiteturas distribuídas na nuvem, oferecendo soluções escaláveis, resilientes e desacopladas. Ao adotar esses padrões, empresas podem integrar sistemas de maneira eficiente, processar grandes volumes de dados e garantir uma comunicação robusta entre serviços.

4. Código Básico da Aplicação

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')
def home():
    return jsonify(message="Hello, World!")

if __name__ == "__main__":
    app.run(debug=True)
```

5. Implementação do Pipeline com GitHub Actions

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
```

```
        python-version: '3.8'

    - name: Install dependencies
      run: pip install -r requirements.txt

    - name: Lint code
      run: flake8 app.py

test:
  runs-on: ubuntu-latest
  needs: build
  steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: '3.8'

    - name: Install dependencies
      run: pip install -r requirements.txt

    - name: Run tests
      run: pytest

deploy:
  runs-on: ubuntu-latest
  needs: test
  steps:
    - name: Deploy to Heroku (ou outra plataforma)
      run: echo "Deploy realizado com sucesso!"  # Substitua pela lógica de deploy
```

Bibliografia

Data 25/09/2024 às 12:00 h

Padrões de Mensagens

https://learn.microsoft.com/en-us/azure/architecture/patterns/category/messaging

Data 26/09/2024 13:00h

Documentação Flask Oficial

https://flask.palletsprojects.com/en/3.0.x/

Data 27/09/2024 15:00

GitHub Actions

https://docs.github.com/en/actions/use-cases-and-examples/deploying/deploying-with-github-actions