Aumentando a confiabilidade da sua aplicação com o Apache Kafka

Gabriel G. da Silva[1]; Anderson Silva do Nascimento (Orientador)

Escola de Ciência e Tecnologia – Universidade do Grande Rio Professor José de Souza Herdy (UNIGRANRIO)

Rua Professor José de Souza Herdy, 1160 - Jardim Vinte e Cinco de Agosto, Duque de Caxias - RJ, 25071-202

gabriel.gomes@unigranrio.br; anderson.nascimento@unigranrio.edu.br

Abstract. *This meta-paper aims to talk about Apache Kafka, demonstrate how it is an excellent tool for messaging architecture and demonstrate how it is extremely useful for robust systems and can provide great reliability in the system. The project idea is to use Apache Kafka as a backup and repository for two services that work together, when one becomes unavailable, Kafka will store the data and when the endpoint comes back, it uses the data that is in the tool and continues its operation.*

Resumo. *Este meta-artigo tem como objetivo falar sobre o Apache Kafka, demonstrar como ele é uma excelente ferramenta para a arquitetura de mensagens e demonstrar como ela é extremamente útil para sistemas robustos e pode proporcionar uma grande confiabilidade no seu sistema. A ideia de projeto é usar o apache Kafka como um backup e repositório para dois serviços que trabalham em conjunto, quando um ficar indisponível, o Kafka irá armazenar os dados e quando o endpoint voltar ele utilizar os dados que estão na ferramenta e continuar sua operação.*

1. Introdução

O Apache Kafka é uma plataforma open source de processamento stream desenvolvida pela Apache Software Foundation. Com o passar dos anos ela foi se modelando para um propósito específico que é servir de um repositório central de fluxo de dados.

O Kafka é ideal para aplicações robustas que precisam lidar com um grande fluxo de processamento de dados e mensagens, o diferencial da ferramenta é que ela armazena seus dados no disco rígido (como fosse log), com isso ele permite a repetição das mensagens e a ferramenta não perde desempenho ao se comparar com as outras ferramentas que armazena as mensagens na memória.

1.2. Objetivo do Projeto

O projeto consiste em criar uma pequena aplicação que simule um sistema de pagamento, ele consiste basicamente em 2 serviços. Um será o sistema de pagamento em si, que fará todo o processamento da conta e gerará a fatura, o outro será o sistema de envio de fatura para o cliente.

Objetivo será derrubar o serviço de envio de fatura e utilizar o Kafka para armazenar os boletos gerados, para que quando o serviço de envio de fatura retorne, ele irá ver as faturas que foram salvos no Kafka e em seguida enviará as faturas para os e-mails indicados.

2. Referencial teórico

Segundo a empresa Confluent[1], que é especializada em big data e foi fundada pela equipe que originalmente criou o apache kafka, o kafka é uma plataforma de processamento de stream capaz de lidar com trilhões de eventos por dia.

Existem diversas empresas que oferecem plataformas que utilizam ou melhoram os recursos do kafka e uma delas é a já mencionada Confluent, oferecendo diversos recursos comerciais e comunitários que visam aprimorar a experiência com o Kafka.

Segundo o Report Apache Kafka 2018[2], o Kafka está cada vez mais sendo adotado pelas empresas, tanto pelas grandes quanto pelas pequenas e o principal crescimento dela é na arquitetura orientada a eventos. De acordo com o mesmo relatório, 94% das empresas pretendem planejam implementar novos aplicativos ou sistemas utilizando o Kafka e também relata que de 6 entre 10 organizações (63%) dependem do Kafka para suas arquiteturas de microsserviços um grande crescimento se comparado no ano relatório passado que foi de 29%.

3. Kafka

O Apache Kafka foi originalmente desenvolvido como um serviço de mensageria pelo Linkedin em 2011, teve seu código aberto no início de 2011 e desde sua criação, ele evolui de uma simples ferramenta de serviço de mensageria, para uma plataforma open-source completa de processamento de stream.

3.1. Arquitetura

O Kafka armazena as mensagens por chave e valor, que vêm de inúmeros processos arbitrários chamado de “producers” ou “produtores”. Os dados podem ser particionados em diferentes “partições” e diferentes “tópicos”. Dentro de uma partição, as mensagens são organizadas pela sua posição e indexados e armazenados juntamente. Os processos chamados de “consumers” ou “consumidores” podem ler as mensagens.

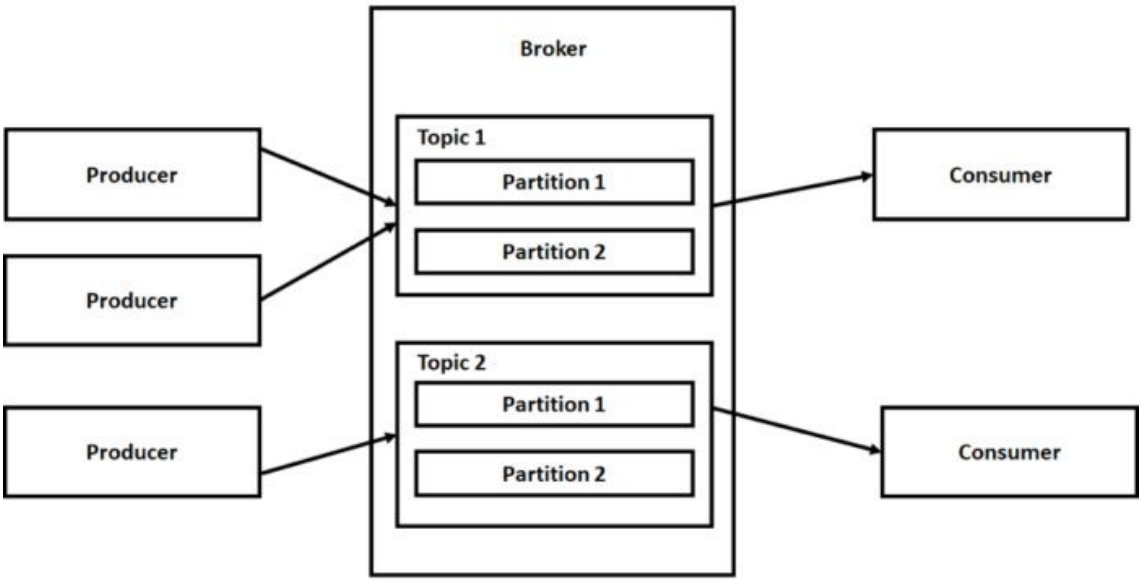

Figure 1. como funciona a arquitetura padrão do kafka

Fonte: Info Word

O kafka também é um banco de dados e as mensagens são armazenadas no disco rígido. Graças a isso, o kafka permite que as mensagens sejam repetidas e a repetição é totalmente configurável, você pode optar por não repetir ou por repetir dentro de prazo.

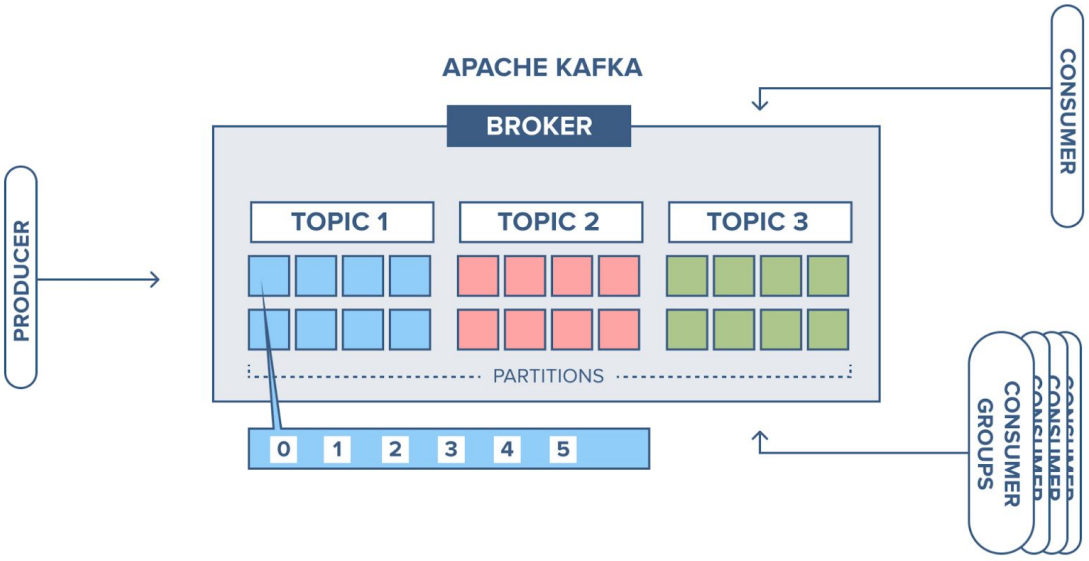

Figure 1. como funciona a arquitetura e a organização de mensagens

Fonte: Blog Cloud Kafka

O importante a se destacar, é que mesmo as mensagens sendo armazenadas no disco rígido (que é mais lento que a memória ram), o Kafka não perde em desempenho ao se comparar com outras ferramentas de mensageria e além disso o Kafka traz a vantagem de repetir as mensagens, coisa que não é possível nas ferramentas que armazenam na memória ram.

3.2. Aplicabilidade

Hoje o Kafka possui diversas aplicabilidades, mas ele é usado principalmente para:

- Mensageria.
- CDC (Change Data Capture).
- ETL em tempo real.
- Rastreamento de atividade Web.
- Agregação de Logs.
- Processamento Stream.

No projeto, usaremos do serviço de mensageria para gerar confiabilidade em uma aplicação, que necessita de dois serviços que trabalham em conjunto.

3.2.1. O'Que é Mensageria?

Em resumo, é basicamente um serviço para intermediar a troca de mensagens entre sistemas.

Por que usar um serviço de mensageria:

- Desacoplamento de serviços
- Garantia de entrega
- Pub/Sub ou P2P

Componentes:

- Canal
- Mensagem
- Endpoint

3.3. Vantagens do apache Kafka

A maior vantagem do Apache Kafka é a escalabilidade, se você possui um projeto robusto ou pensa em crescer cada vez mais a sua aplicação, você deveria considerar o uso do Kafka ou outras ferramentas de mensageria, basta avaliar a vantagem de cada uma e ver qual se adequa mais a sua aplicação. Outra vantagem do Kafka é a tolerância a falhas, como já mencionado, o Kafka armazena as suas mensagens no disco rígido (como fosse um log), com isso as mensagens podem se repetir quantas vezes você quiser, sendo assim, caso ocorra algum problema na rede ou uma falha técnica, as mensagens não serão perdidas.

4. Projeto

A ideia do projeto é demonstrar como o Apache Kafka é uma plataforma robusta e com diversas aplicabilidade, o projeto consiste em usar o Apache Kafka como uma ferramenta de confiabilidade, para dois serviços que trabalham em conjunto.

Para isso foi criado dois projetos em .net core, o primeiro é uma Api de pagamento que receberá e validar dados de um pagamento, em seguida ele gerará uma fatura que será encaminhada para o serviço de envio de e-mail que está preparado para receber uma fatura e em seguida, encaminhará para o e-mail cadastrada na mesma. O problema é que como os serviços dependem um do outro para o outro, caso uma fique fora do ar toda a aplicação ficará indisponível e é aí que o Kafka se encaixa, ele servirá como um repositório ou banco de dados de transações caso a Api de enviar e-mail caia.

O serviço de pagamento será o Produtor do Kafka, quando ele não conseguir se comunicar com o serviço de envio de fatura, ele irá encerrar a operação e encaminhará o boleto como mensagem para partição de pagamento que foi criada no Kafka e completará sua operação. O serviço de envio de e-mail será o consumidor, ele enviará todas as faturas que recebeu para o e-mail informado pelo usuário durante o pagamento, caso esse serviço fique fora do ar, ao retornar a sua operação ele irá no kafka e verificará todas as mensagens que não foram consumidas e enviará todas as faturas encontradas.

4.1 Ferramentas Utilizadas

Para iniciar os serviços do Kafka e do Zooker, foi configurado um Docker-Compose que conterá todas as configurações necessárias do Zookeeper e Kafka.

4.1.1 Docker e Docker compose

O Docker é utilizado para iniciar um container com Kafka e o ZooKeeper e o Docker Compose é o arquivo que conterá as configuração necessárias, como por exemplo, como o Kafka e o ZooKeeper se comportará quando for iniciado.

4.1.2 Apache Kafka e ZooKeeper

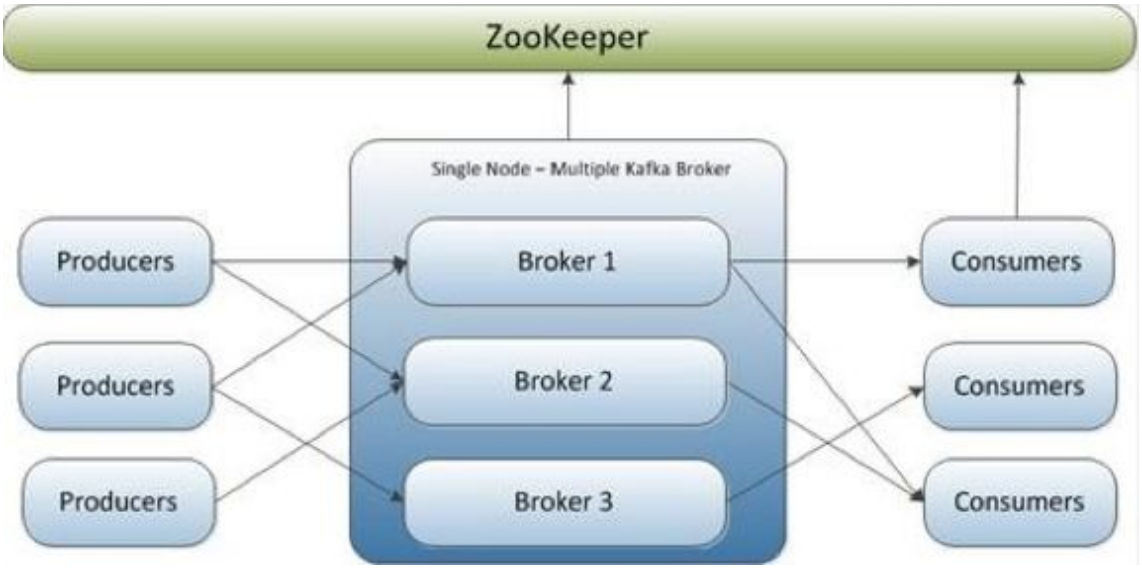

Figura 1. Papel do ZooKeeper no Apache Kafka

Fonte: Learning Apache Kafka - Second Edition

O Kafka será o repositório de todas as faturas que foram geradas no serviço de pagamento caso o serviço de envio de e-mail esteja fora do ar, para isso foi criada uma partição com o nome “payment” que será armazenado.

O Zookeeper[4] é basicamente o responsável por supervisionar o Kafka, ele é responsável pela configuração e sincronização de diferentes Clusters no Kafka.

4.1.3 .Net e C#

Os serviços de Pagamento e Envio de e-mail foram criados em .NET, que é um framework livre e de código aberto, ambos os serviços foram escritos com a linguagem C#.

5. Resultados

O Kafka é uma plataforma robusta e com diversas possibilidades e o entendimento do seu funcionamento é respectivamente simples, a maior dificuldade dele é justamente as suas questões técnicas. A criação de Clusters, tópicos, partições, produtores e consumidores, são bem confusas no começo e o método de iniciar o Kafka e o ZooKeeper é pior ainda, no entanto, existem diversas ferramentas que facilitam esse procedimento. Como já citado neste artigo, a Confluent possui diversas ferramentas e tutoriais que facilitam muito o desenvolvimento, e uma delas que ajudou muito no desenvolvimento desse projeto foi a sua biblioteca para o .NET, que facilita muito o trabalho com produtores e consumidores, além disso, ela possui uma documentação que explica do zero a criação de produtores e consumidores em sua biblioteca em .NET.

Como o desenvolvimento foi possível notar o grande potencial do Kafka e reconhecer o por que diversas empresas no mundo estão adotando ele. Graças ao Kafka, nenhuma fatura foi perdida quando o segundo serviço ficou fora do ar e foi reparado que ao invés de usar o Kafka como um backup, ele poderia ser usado por toda a aplicação, cortando assim custo de desenvolvimento, evitar soluções repetitivas e a possibilidade de repetir implementações e usar o Kafka em diferentes projetos.

O único ponto negativo que foi encontrado durante o desenvolvimento, foi que na biblioteca da Confluent, foi que o consumidor do Kafka trabalha apenas com requisições síncronas, sendo assim, caso não houvesse mensagens no Kafka, o consumidor fica esperando até que uma mensagem seja armazenada, parando assim toda a aplicação sendo necessário um estudo para contornar esse problema e demonstrando assim, que ao invés de usar o Kafka apenas como backup, o maior ganho com ele seria utilizá-lo em todo o serviço.

6. Conclusão

É de conhecimento que a época do ano que as lojas mais lucram é na Black Friday e também é de conhecimento que as mesmas investem muito principalmente em infraestrutura para manter suas aplicações funcionando perfeitamente nessa época, porque cada minuto que um serviço fica fora do ar é milhares de reais perdido.

A objetivo desse artigo foi demonstrar como o Apache Kafka pode ser uma excelente plataforma para suprir essa necessidade, além de ser uma plataforma robusta para aplicações com grande escalabilidade, Kafka e suas API’s possuem diversas

aplicabilidades e simplificam a criação de aplicativos orientado a dados e o gerenciamento de sistemas de back-end complexos, por isso essa plataforma é utilizada por diversas empresas no mundo.

Se você possui uma aplicação que consome diversos microsserviços, principalmente de empresas terceirizadas, e você percebeu que a aplicação está um pouco lenta ou sente que com o tempo ela perderá desempenho, você deveria considerar considerar o uso do Apache Kafka na intermediação desses microsserviços. Como vimos no projeto apresentado nesse artigo, o Apache Kafka pode servir como uma grande ferramenta de confiabilidade para sua aplicação.

Códigos utilizados no projeto estão disponíveis via GitHub no link: https://github.com/gabrielgs44/TCC-Apache-Kafka

Referências

[1] Confluent. Entenda oque é o Apache Kafka. Disponível em: https://www.confluent.io/what-is-apache-kafka/ Acessado em: 1 de Novembro de 2020.

[2] Confluent. Report Apache Kafka. Disponível em: https://www.confluent.io/apache-kafka-report/ Acessado em: 30 de Novembro de 2020.

[3] Apache Foundation. Apache ZooKeeper. Disponível em:

https://zookeeper.apache.org/ Acessado em: 30 de Novembro de 2020.