TREINAMENTO APACHE KAFKA

EMENTA

- Introdução ao Apache Kafka
- Sistemas de mensageria
- Mensageria com o Apache Kafka
- Arquitetura do Apache Kafka
- Fluxo de trabalho no Apache Kafka
- Principais componentes do Apache Kafka
- API do Apache Kafka
- Casos de uso para Apache Kafka
- RabbitMQ x Kafka
- Exemplo de código / implementação

Introdução ao Apache Kafka

- Apache Kafka é uma solução de software baseada em um processo de streaming. É um sistema de mensagens do tipo publicação/assinatura(publish/subscriber) que permite a troca de dados entre aplicativos e servidores.

- O Apache Kafka foi originalmente desenvolvido pelo LinkedIn em 2010 e, posteriormente, doado à Apache Software Foundation. Atualmente, ele é mantido pela Confluent sob a Apache Software Foundation.

- No ano de 2011 Kafka tornou-se um projeto público (licenciado pela Apache License).

- O Kafka funciona bem como um substituto para um barramento de mensagens mais tradicional (IBM MQ).

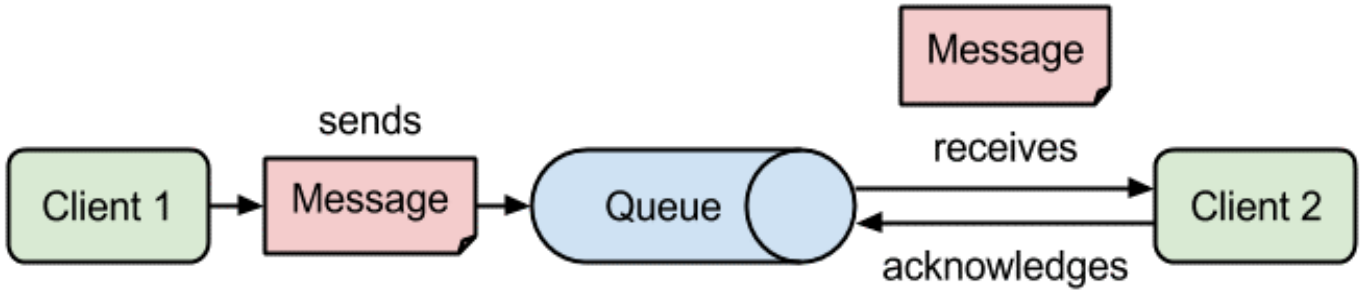
- O Apache Kafka resolveu o problema letárgico da comunicação de dados entre o remetente e o destinatário.

Sistemas de Mensageria

Existem dois tipos de Sistema de Mensagens:
- **Ponto-a-Ponto**
- **Publicação (Publish) / Assinatura (Subscriber)**

1. Ponto a Ponto
As mensagens são mantidas em uma fila, mas uma determinada mensagem pode ser consumida por no máximo um consumidor apenas. Depois que um consumidor lê uma mensagem na fila, ela desaparece dessa fila.

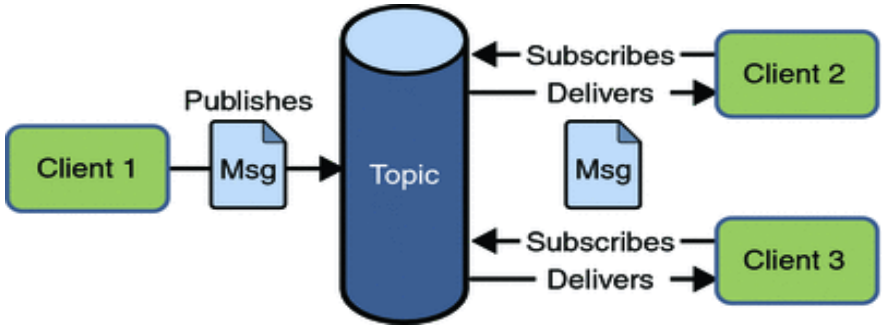
O exemplo típico desse sistema é um Sistema de Processamento Ordenado/Enfileirado, em que cada pedido será processado por um processo, mas vários processos de consumo também podem funcionar ao mesmo tempo. O diagrama a seguir descreve a estrutura:



2. Sistema Publicação/Assinatura(Publish/Subscribe).

As mensagens são persistidas em um tópico. Ao contrário do sistema ponto-a-ponto, os consumidores podem se inscrever(assinar) em um ou mais tópicos e consumir todas as mensagens desse tópico. No sistema Publicação/Assinatura(Publish-Subscriber), os produtores de mensagens são chamados de publicadores e os consumidores de mensagens são chamados de assinantes.

Um exemplo na vida real é o Youtube, que publica diferentes canais como, comédia, entrevista, música, etc., e qualquer pessoa pode assinar(e Ativar o Sino) seu próprio conjunto de canais e obtê-los sempre que seus canais assinados estiverem disponíveis ou lançarem algum conteúdo novo.

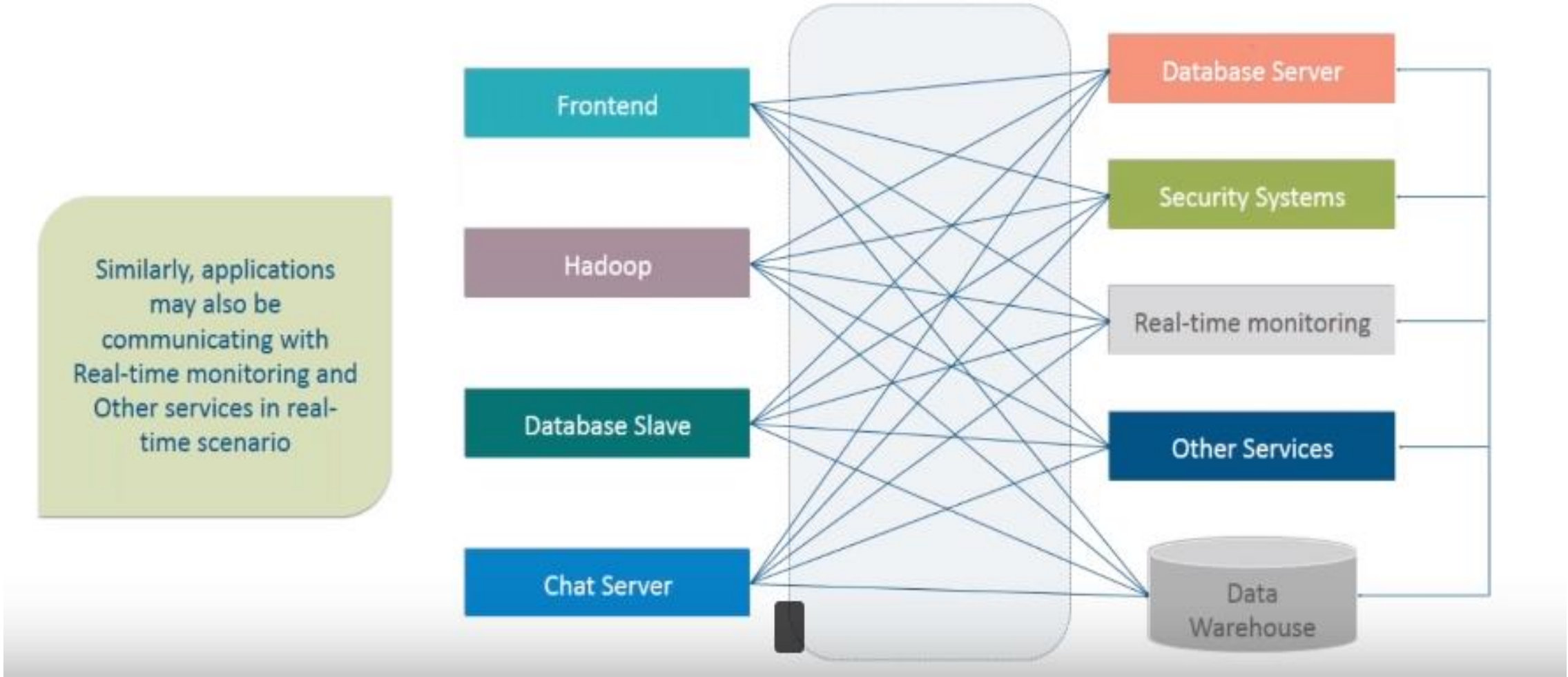


APACHE kafka™
A distributed streaming platform

pix powered by Banco Central

stefanini

Sistemas de mensageria complexos (Sem auxílio do Kafka)

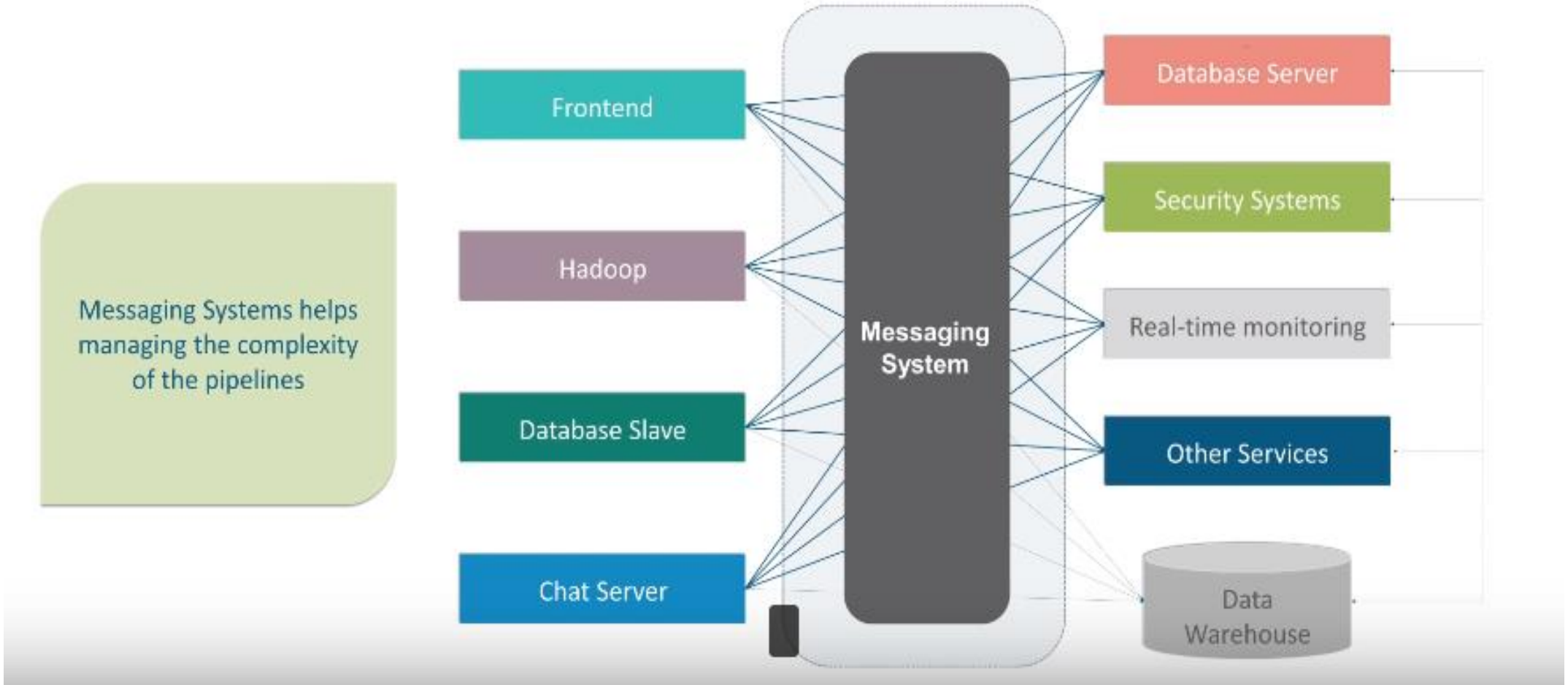


APACHE kafka™
A distributed streaming platform

pix powered by Banco Central

stefanini

Sistemas de mensageria complexos (Com auxílio do Kafka)



Arquitetura Apache Kafka

- Kafka é um sistema de mensagens/logs distribuído e replicado de mensagens. O Kafka não tem o conceito de fila, o que pode parecer estranho à primeira vista, já que, é usado principalmente como um sistema de mensagens. As filas(queues) são sinônimos de sistemas de mensageria há muito tempo.

- Vamos analisar um pouco o “sistema de mensagens/logs distribuído e replicado de mensagens”:

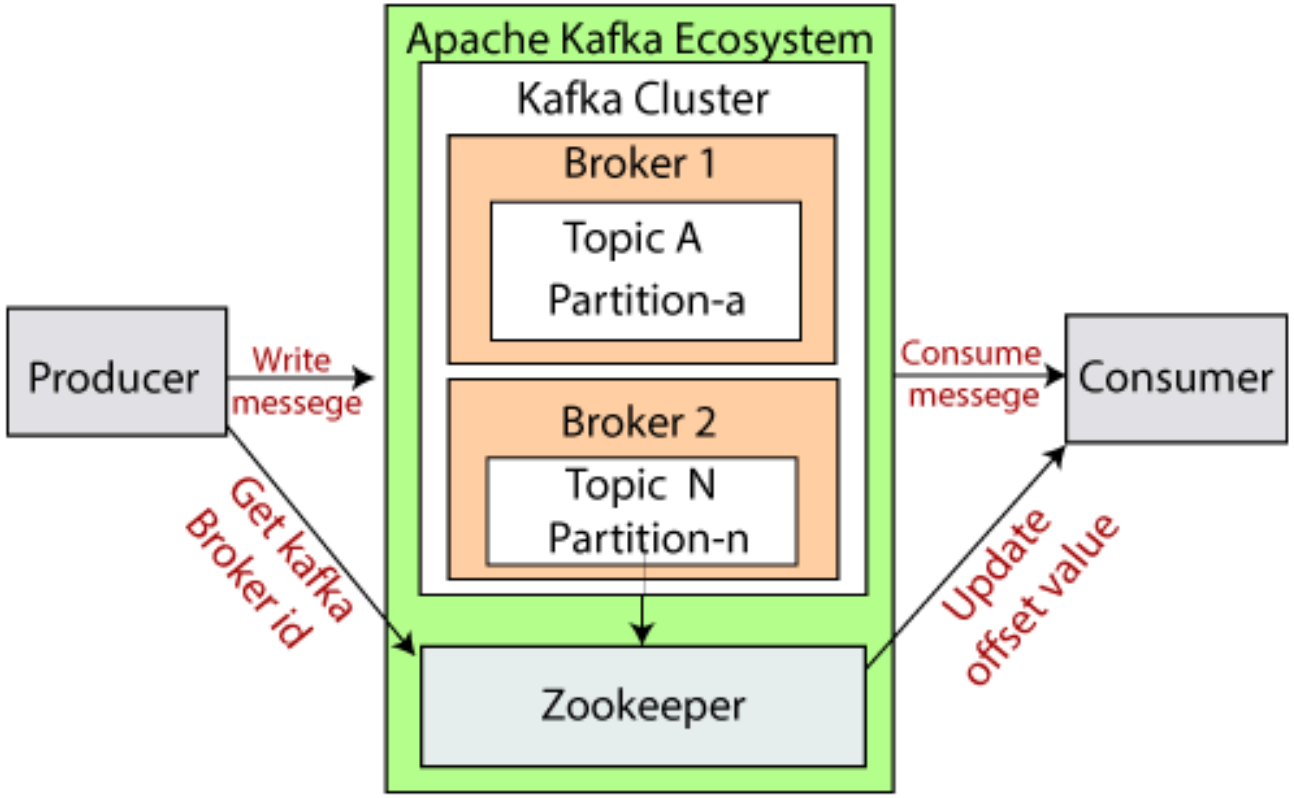
    1. Distribuído porque o Kafka é implantado como um cluster de nós, para tolerância a falhas e escalanomento.

    2. Replicado porque as mensagens geralmente são replicadas(replication factor) em vários nós/brokers(servidores).

    3. O Kafka é tão poderoso em relação a performance e escalabilidade que permite você lidar com o fluxo contínuo de mensagens.

    4. Confirmação de persistência (commit log ack), já que as mensagens são armazenadas em tópicos/partições/logs. Esse conceito de partição ou log é o principal diferencial do Kafka.

Apache Kafka como um Sistema de Mensageria



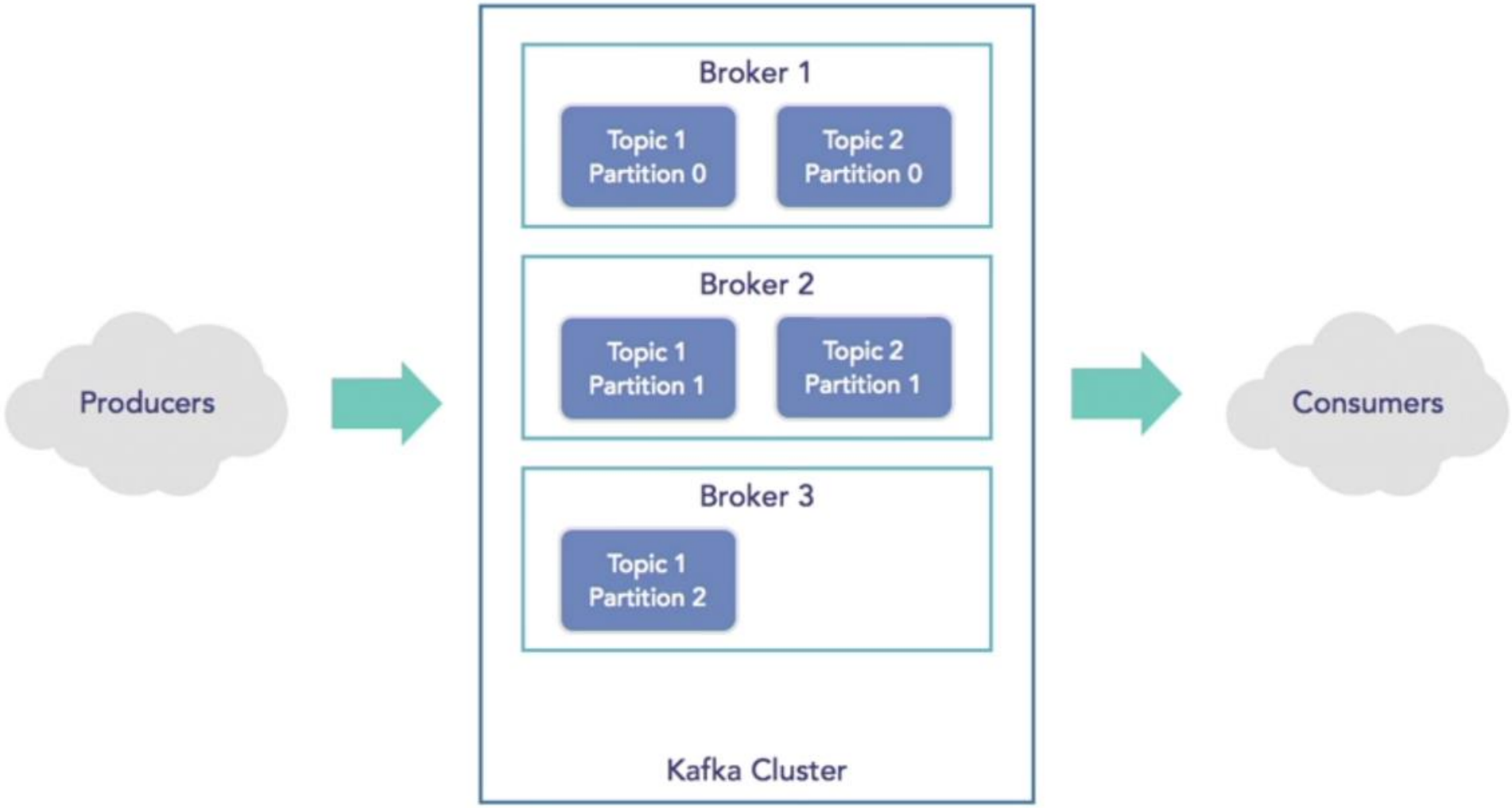
Apache Kafka Architecture

APACHE kafka
A distributed streaming platform

pix
powered by Banco Central

stefanini

Arquitetura Apache Kafka



Qual é o caminho de uma mensagem no Kafka

O fluxo de uma mensagem assim que, chega ao kafka, poderia ser resumido nos seguintes pontos:

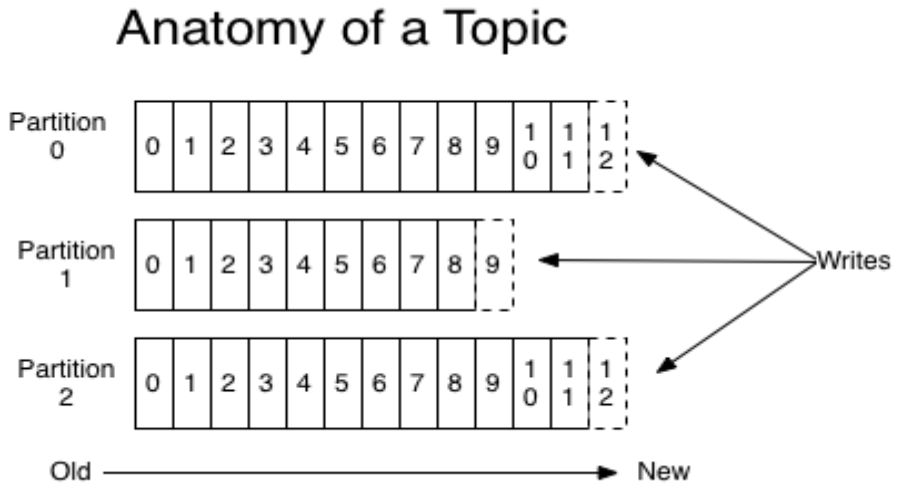
- Os produtores enviam mensagens para um tópico em intervalos regulares.
- O broker Kafka armazena todas as mensagens nas partições configuradas para esse tópico específico. Isso garante que as mensagens sejam compartilhadas igualmente entre as partições. Se o produtor enviar duas mensagens e houver duas partições, o Kafka armazenará uma mensagem na primeira partição e a segunda na segunda partição.
- O consumidor assina um tópico específico.
- Assim que o consumidor assina um tópico, Kafka fornecerá o deslocamento/offset(faixa da música) atual do tópico para o consumidor/assinante/subscriber e também guardará o offset/deslocamento no Zookeeper.
- O consumidor solicitará ao Kafka em um intervalo regular (como 100 ms) quaisquer novas mensagens.
- Assim que o Kafka recebe as mensagens dos produtores, ele já as encaminha para os consumidores.
- O consumidor receberá a mensagem e a processará.
- Assim que as mensagens forem processadas, o consumidor enviará uma confirmação(ack) ao broker Kafka.
- Depois que o Kafka recebe uma confirmação(ack), ele altera o deslocamento/offset para o novo valor e o atualiza no Zookeeper. Uma vez que os deslocamentos são mantidos no Zookeeper.
- Este fluxo acima se repetirá até que o consumidor interrompa/conclua a solicitação/consumo de mensagens.
- O consumidor tem a opção de retroceder/pular para o deslocamento/offset desejado de um tópico a qualquer momento e ler todas as mensagens subsequentes que por ventura já tenham sido processadas.

Componentes do Kafka: Tópico

Usando os seguintes componentes, o Kafka consegue mensagens:

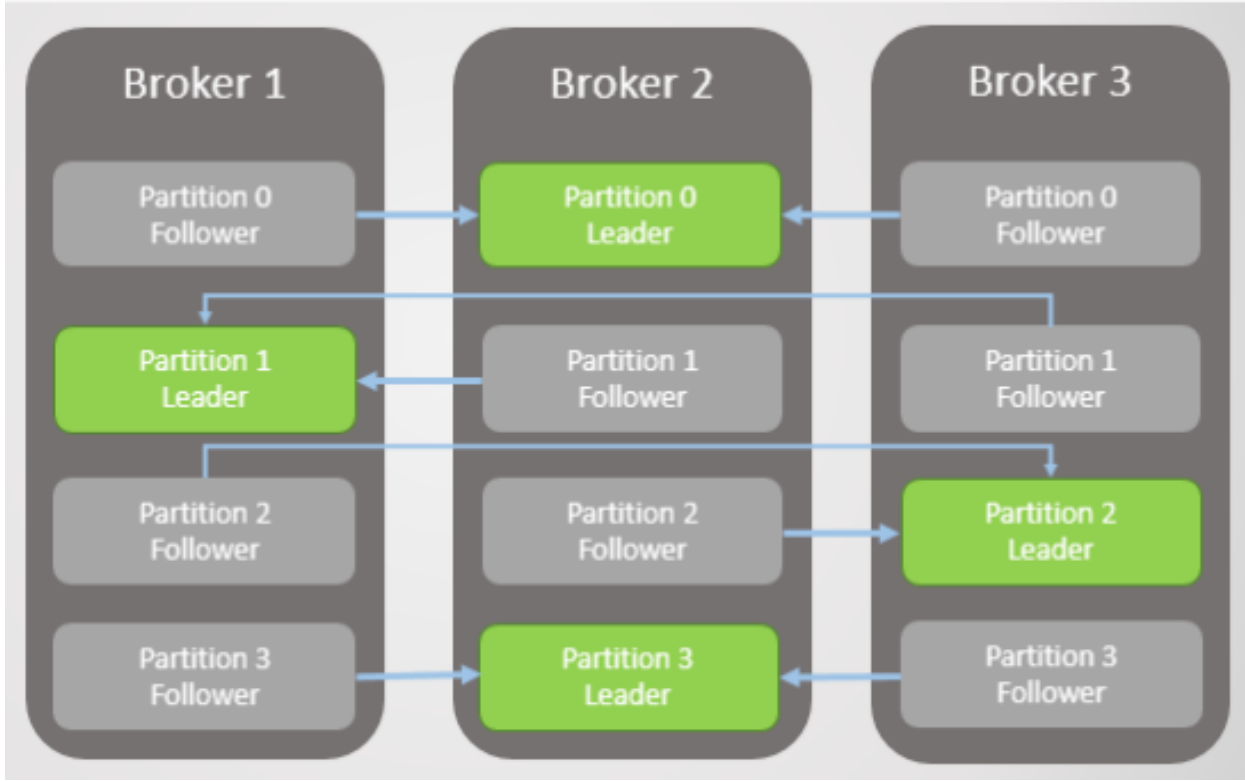
1. Tópico
Basicamente, o Tópico é um nome exclusivo para Kafka Stream. O Tópico é um componente no qual os registros são publicados guarda mensagens. Os tópicos no Kafka são sempre com vários assinantes/consumidores; ou seja, um tópico pode ter zero, um ou muitos consumidores que assinam os dados gravados nele.

Cada partição é uma sequência ordenada e **imutável** de registros continuamente incrementada a um log de commit/ack estruturado. Cada um dos registros nas partições é atribuído a um número de identificação sequencial denominado **OFFSET**, que identifica exclusivamente cada registro na partição.



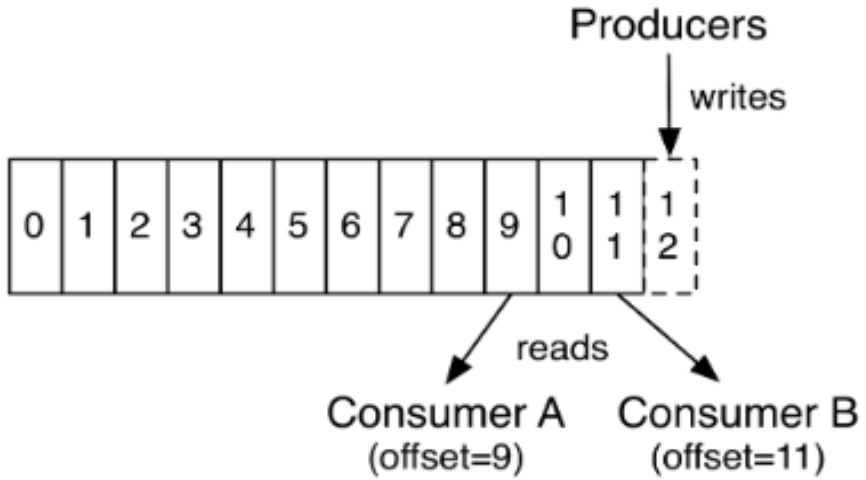
Componentes Kafka: Partições

- As partições para um mesmo tópico são distribuídas em vários brokers no cluster.
- As partições são replicadas em vários servidores; o número de réplicas é um parâmetro configurável.
- Cada partição tem sempre apenas um servidor como líder e vários servidores como seguidores.
- Cada servidor atua como líder para algumas de suas partições e como seguidor de algum outro.
- Os produtores são responsáveis por escolher quais mensagem atribuir a qual partição, dentro de cada tópico com base na chave atribuída à mensagem e a sua partição lider.



Kafka Components

- O cluster Kafka persiste de forma durável todos os registros publicados - tenham ou não sido consumidos - usando um período de retenção configurável. Por exemplo, se a política de retenção for definida para dois dias, nos dois dias após a publicação de um registro, ele ainda estará disponível para consumo, após 1 minuto desse tempo o mesmo será descartado para liberar espaço.
- O desempenho do Kafka é efetivamente constante em relação ao tamanho dos dados, portanto, armazenar dados por um longo tempo não é um problema. Esta é uma das maiores diferenças entre RabbitMQ / ActiveMQ e o Kafka.

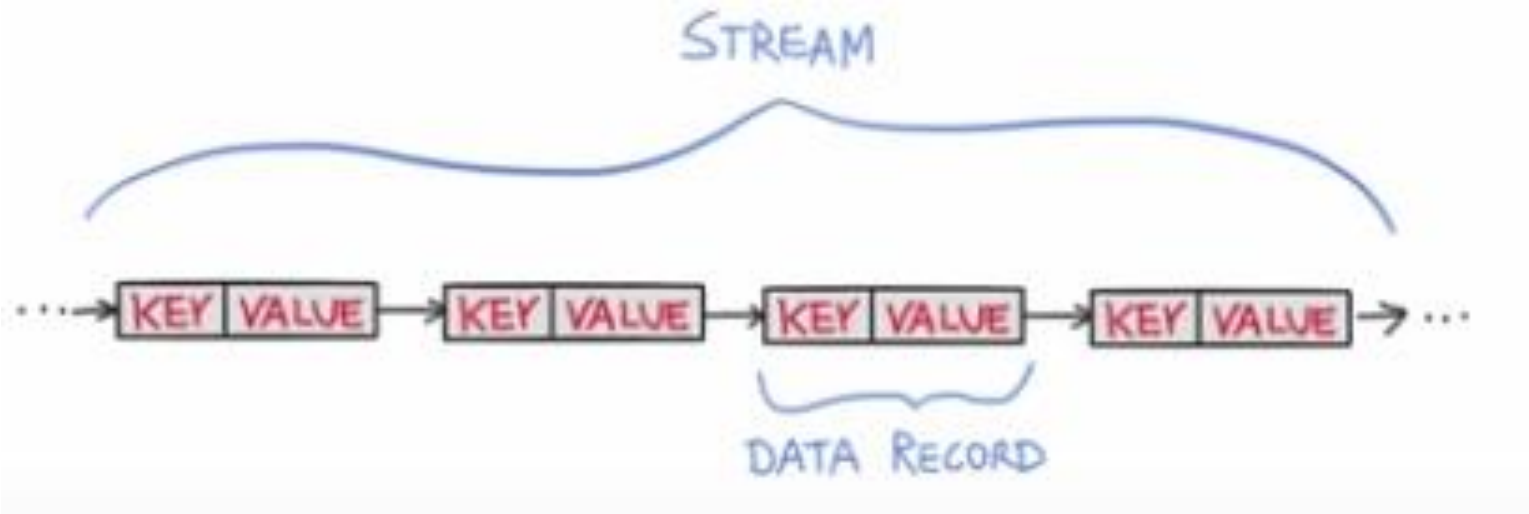


Componentes do Kafka: O que é um stream de dados ?

Pense em Stream como um infinito. Um fluxo contínuo de dados em tempo real, onde os dados são pares de valores-chave.

Na API de stream do Kafka, transforma-se e aumenta os dados.

- Suporte a processamento de stream por registro com milissegundo.



Resumo dos componentes.

- **Kafka Cluster:** Um cluster Kafka é um sistema composto por diferentes brokers, tópicos e suas respectivas partições. Os dados são gravados no tópico dentro do cluster e lidos pelo próprio cluster.
- **Producers:** Um produtor envia ou grava dados / mensagens no tópico dentro do cluster. Para armazenar uma grande quantidade de dados, diferentes produtores em um aplicativo enviam dados para o cluster Kafka.
- **Consumers:** Consumidor é aquele que lê ou consome mensagens do cluster Kafka. Pode haver vários consumidores consumindo diferentes tipos de dados do cluster. A beleza do Kafka é que cada consumidor sabe de onde precisa consumir os dados.
- **Brokers:** Um servidor Kafka é conhecido como Broker. Um Broker é uma ponte entre produtores e consumidores. Se um produtor deseja gravar dados no cluster, eles são enviados ao servidor Kafka. Todos os Brokers estão dentro do próprio cluster Kafka. Além disso, pode haver vários Brokers.
- **Topics:** É um nome comum ou um título fornecido para representar um tipo semelhante de dados. No Apache Kafka, pode haver vários tópicos em um cluster. Cada tópico especifica diferentes tipos/padrões de mensagens.
- **Partitions:** Os dados ou mensagens são divididos em pequenos pedaços, conhecidos como partições. Cada partição carrega dados dentro de si com um valor de offset/marcador. Os dados são sempre gravados de forma sequencial. Podemos ter um número infinito de partições com valores de offset/marcador infinitos. No entanto, não é garantido em qual partição a mensagem será gravada.
- **ZooKeeper:** O ZooKeeper é usado para armazenar informações sobre o cluster Kafka e detalhes dos clientes consumidores. Ele gerencia Brokers mantendo uma lista deles. Além disso, um ZooKeeper é responsável por escolher um líder para as partições. Se ocorrer alguma mudança, como dados do Broker, novos tópicos, etc., o ZooKeeper enviará notificações ao Apache Kafka. Um ZooKeeper é projetado para operar com um número ímpar de servidores Kafka. Zookeeper tem um servidor líder que gerencia todas as gravações dentro do cluster, e o resto dos servidores são os seguidores que tratam de todas as leituras. No entanto, um usuário não interage diretamente com o Zookeeper, mas por meio dos Brokers Kafka. Nenhum servidor Kafka pode ser executado sem um servidor zookeeper (até a versão 2.8 do kafka). É obrigatório executar o servidor zookeeper.

Casos de uso para o Kafka

Existem vários casos de uso do Kafka que mostram porquê realmente usamos o Apache Kafka.

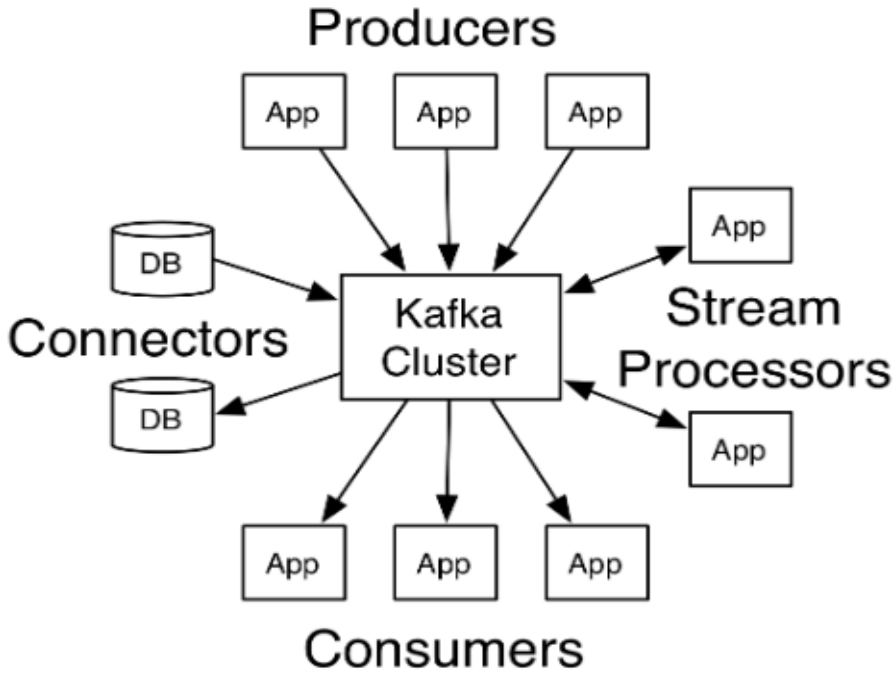
Mensageria:
Para uma solução de mensagens mais tradicional, o Kafka funciona bem como um substituto. Podemos dizer que o Kafka tem melhor rendimento, particionamento integrado, replicação e tolerância a falhas, o que o torna uma boa solução para aplicativos de processamento de mensagens em grande escala.

Métricas:
Para dados de monitoramento operacional, o Kafka disponibiliza um bom conjunto de ferramentas. Inclui estatísticas agregadas de aplicativos distribuídos para produzir métricas centralizadas para os seus dados operacionais.

Fornecedor de eventos
Uma vez que o Kafka suporta dados de logs/partições armazenados muito grandes, isso significa que o Kafka é um excelente back-end para aplicativos de origem de eventos.

API principal do Apache Kafka

- A Producer API permite que um aplicativo publique um flux/stream de registros/dados para um ou mais tópicos do Kafka.
- A Consumer API permite que um aplicativo se inscreva/subscribe/consuma em um ou mais tópicos e processe o fluxo/stream de registros produzidos para eles.
- A Streams API permite que um aplicativo atue como um processador de fluxo/stream, consumindo uma stream de entrada de um ou mais tópicos e produzindo um fluxo/stream de saída para um ou mais tópicos de saída, transformando efetivamente(em tempo de execução) as stream de entrada em stream de saída.
- A Connector API permite a construção e execução de produtores ou consumidores reutilizáveis que conectam os tópicos do Kafka a aplicativos ou sistemas de dados existentes. Por exemplo, um conector para um banco de dados relacional pode capturar todas as alterações em uma tabela.



Números do Kafka

Taxa de desempenho

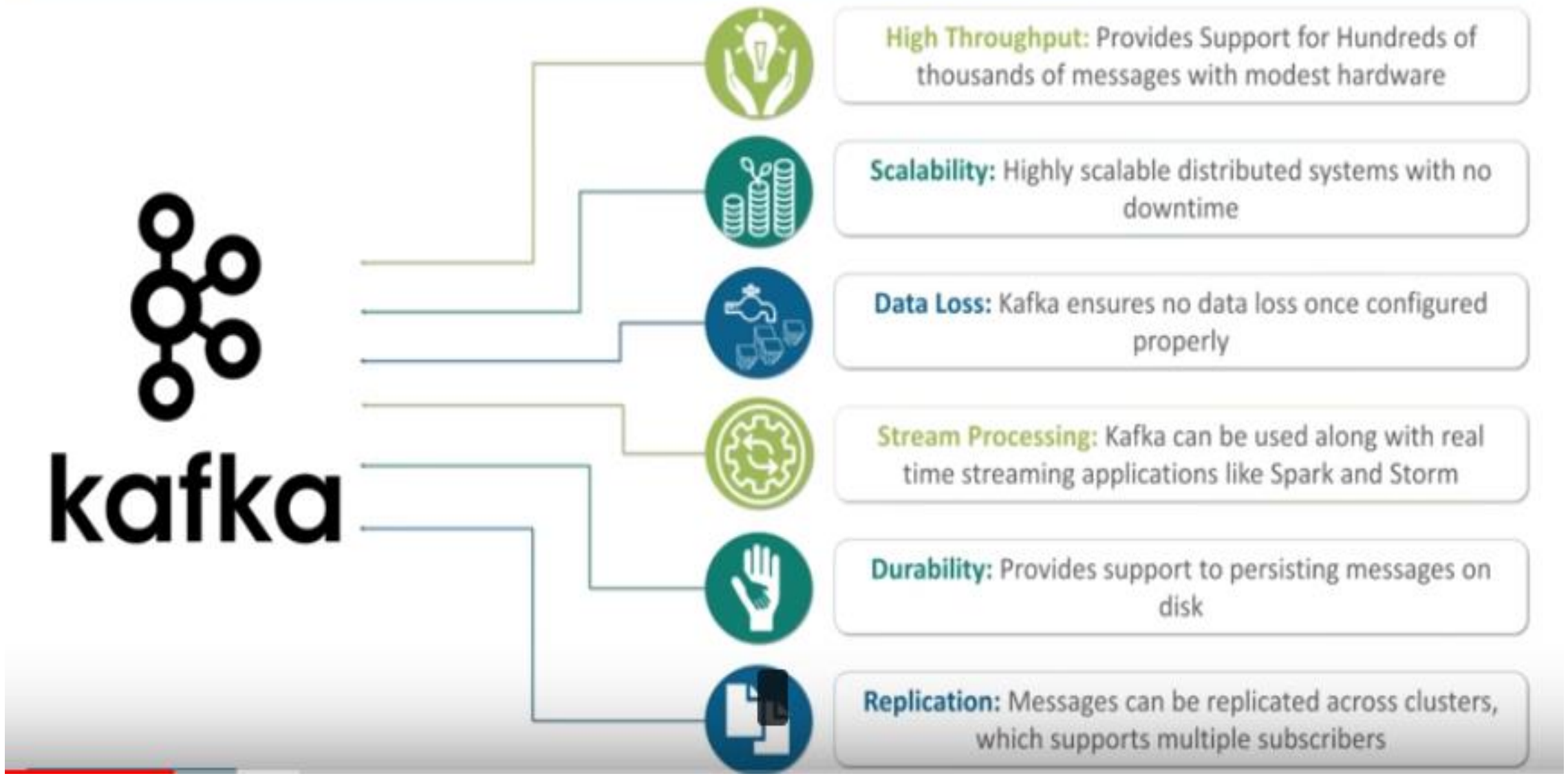
Sua taxa de desempenho é elevada, chegando a 100.000 mensagens / segundo, sem ter impacto na performance / tempo de resposta para os seus consumidores/produtores de mensagens.

Mais testes realizados com o Kafka que demonstram o seu potencial:
https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines

APACHE kafka™
A distributed streaming platform

pix
powered by Banco Central

stefanini



APACHE kafka™
A distributed streaming platform

pix
powered by Banco Central

stefanini