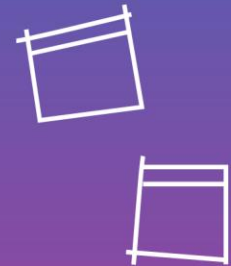




 **VEM SER**  
**DBC**



# Apache Kafka

Aula 1 – Fundamentos + Primeira Mensagem



# Conteúdo do Módulo 4

- Apache Kafka (2 aulas e 1/2)
- Scheduler com SpringBoot (1/2 aulas)
- TF7 (Kafka + Scheduler)

# Conteúdo do Kafka

- Conceitos Fundamentais
- Mensagens
- Arquitetura
- Casos de Uso
- Configuração
- Produtores / Consumidores
- Objetos nas Mensagens
- Utilização de Partições

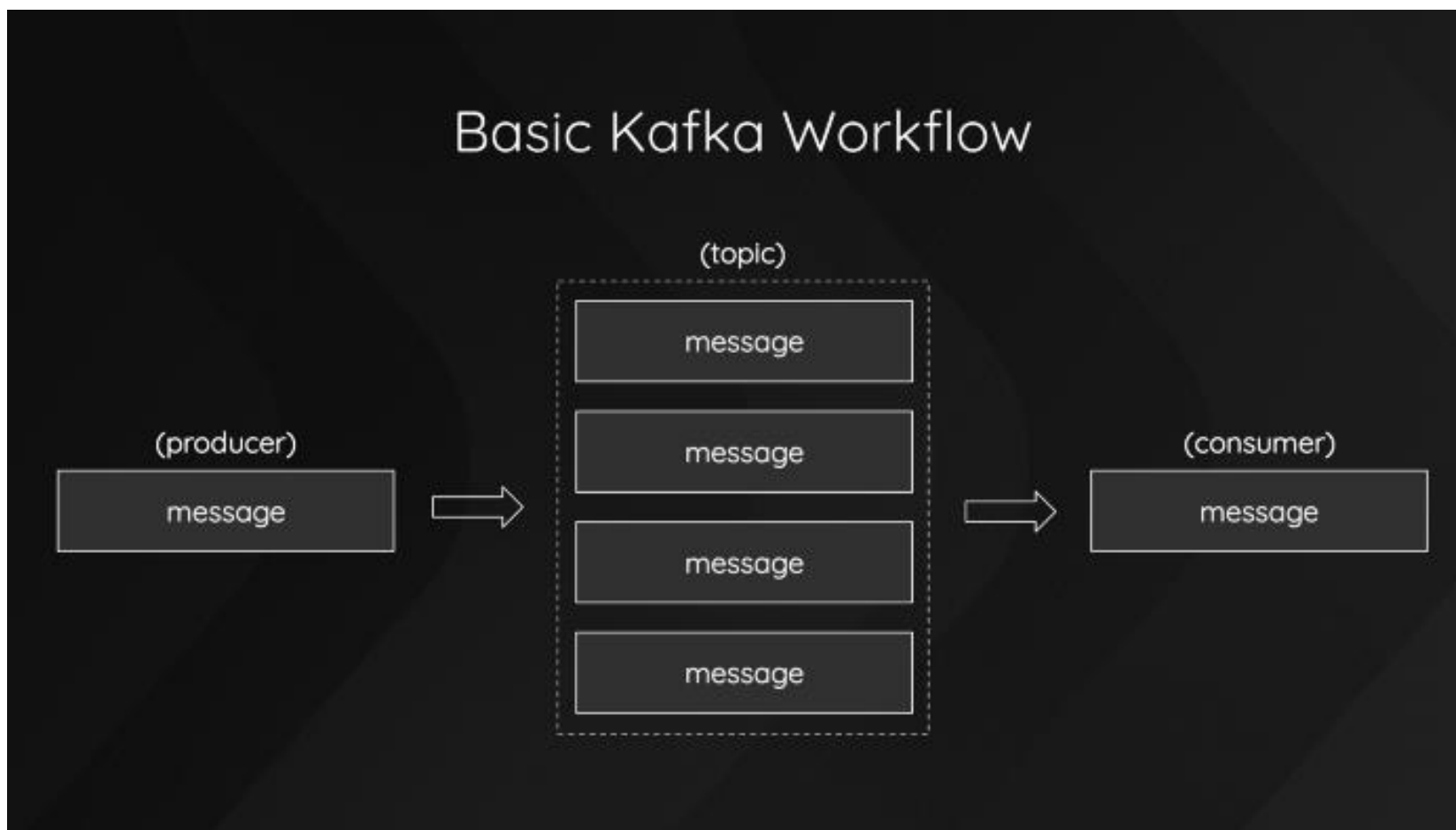
# Conteúdo da Aula

- Conceitos Fundamentais
- Mensagens
- Arquitetura
- Casos de Uso
- Configuração
- Produtores / Consumidores

# O que é o Apache Kafka?

- Apache Kafka é uma plataforma distribuída de mensagens e streaming.
- Consiste em **servidores** e **clientes** que se comunicam por meio de um protocolo de rede TCP.
- O fluxo básico do Kafka pode ser resumido em três ações bem simples:
  - Você **produz** uma mensagem.
  - Essa mensagem é **anexada** em um tópico.
  - Você então **consome** essa mensagem.

# Fluxo Básico



# Conceitos

- Mensagem
- Tópicos
- Producer
- Consumer
- Apache Zookeeper
- Kafka Brokers | Kafka Clusters

# Mensagens

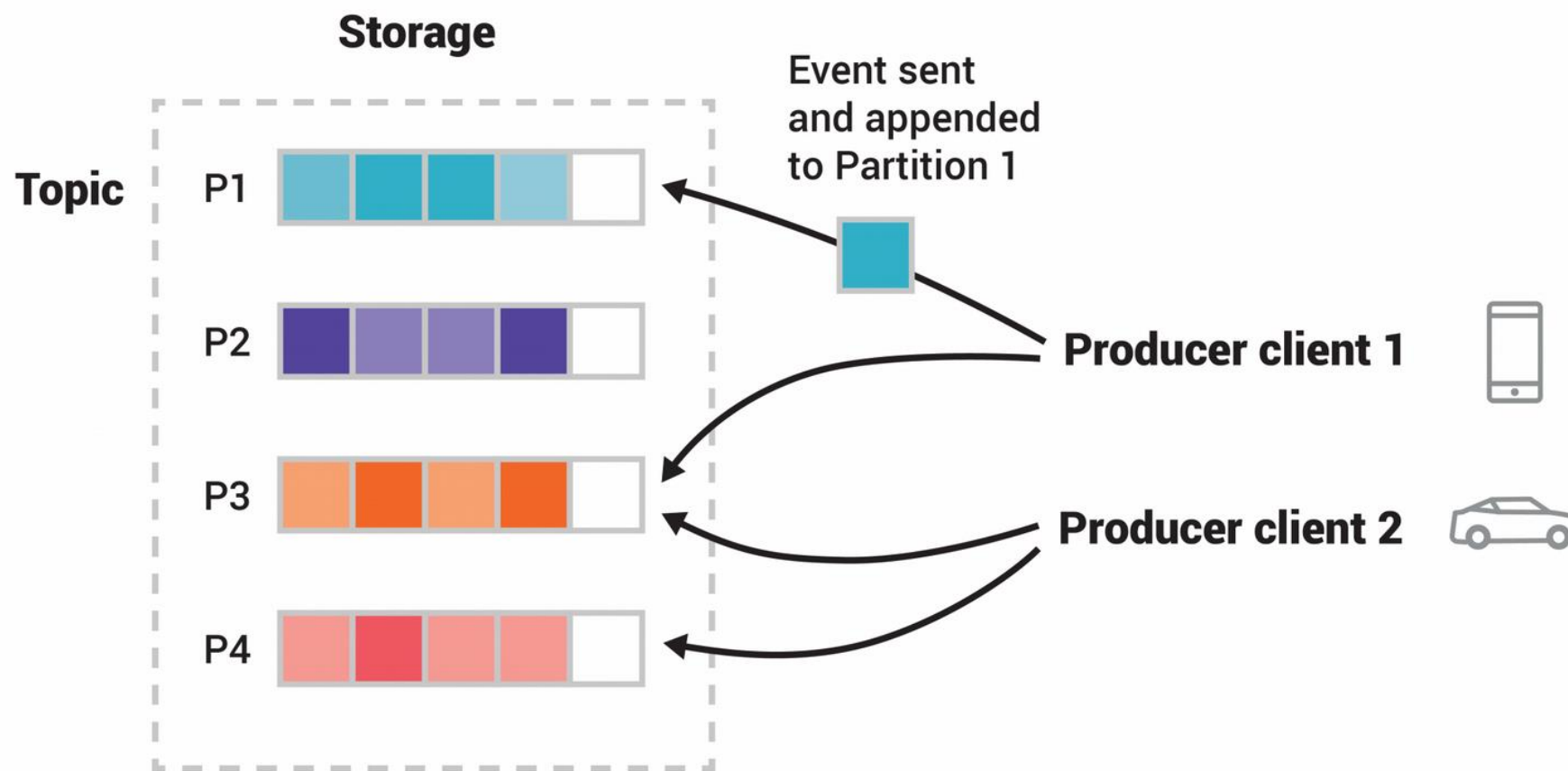
- Principal recurso do Kafka, Todos os eventos do Kafka podem ser resumidos em mensagens.
- Podem ser textual (strings)
- Podem ser definidos Schemas por meio do Avro
- Podem ser composta por chave (key/value)
- Mensagens são imutáveis e ordenadas.



# Tópicos

- Grupo de mensagens dentro do Kafka
- Todas as mensagens ficam dentro de um tópico.
- Para manter a ordenação em um ecossistema de Kafka, os tópicos possuem partições e fatores de replicação.
- Em tópico pode possuir n partições, mas ao receber uma nova mensagem o Kafka automaticamente direciona aquela mensagem para uma partição específica dependendo de sua chave (key).
- Assim mensagens de uma mesma chave estarão apenas em uma única partição, garantindo assim a leitura ordenada de todas as mensagens de um tópico.

# Tópicos



# Producer

- É responsável por enviar uma mensagem para um tópico específico.
- De forma simples, você pode produzir uma mensagem em um tópico.
- Uma vez que uma mensagem é produzida em um tópico o próprio Kafka organiza a mensagem em uma partição, garantindo sempre a ordem das mensagens produzidas.

# Consumer

- Temos os tópicos, e as mensagens dentro dos tópicos. Com o Kafka Consumer é possível ler essas mensagens.
- Ao ler uma mensagem com o consumer, a mensagem não é retirada do tópico.
- Podemos ter vários Kafka Consumers conectados em um mesmo tópico, e cada um terá a posição onde parou de ler.
- Podemos também pode escolher ter vários consumers lendo o mesmo tópico e na mesma partição, para escalar sua aplicação por exemplo, neste caso estes consumers fariam parte de um Consumer Group, e compartilharão sempre a posição final de leitura entre eles (offsets).

# Apache Zookeeper

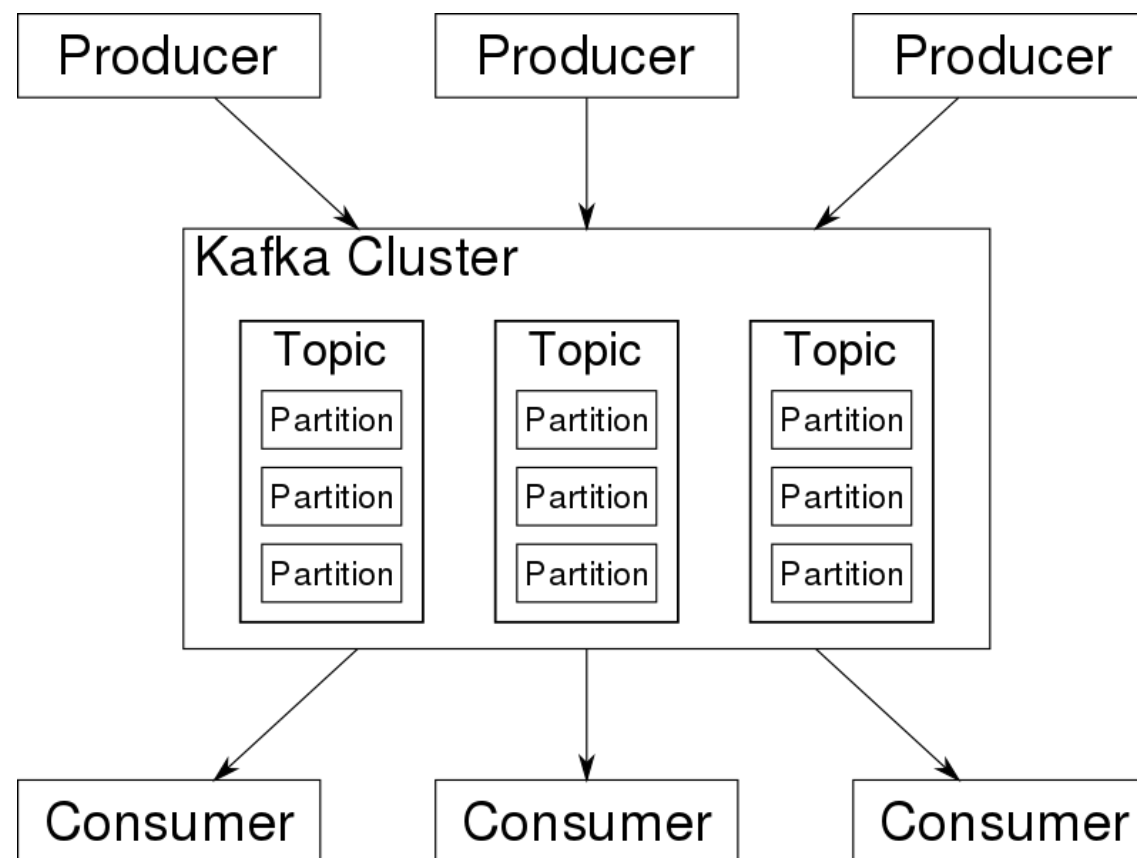
- O Zookeeper é um serviço centralizado para, entre outras coisas, coordenação de sistemas distribuídos.
- O Kafka é um sistema distribuído, e consequentemente delega diversas funções de gerenciamento e coordenação para o Zookeeper.
- Eles possuem uma dependência muito forte, mas isso não é tão ruim.
- O Kafka pode fazer o que ele intencionalmente tem que saber fazer de melhor, delegando essas demais funcionalidades para quem sabe fazer isso bem, sem precisar reinventar a roda.

# Kafka Brokers | Kafka Clusters

- O Broker é o coração do ecossistema do Kafka. Um Kafka Broker é executado em uma única instância em sua máquina. Um conjunto de Brokers entre diversas máquinas formam um Kafka Cluster.



# Arquitetura



# Casos de Uso

- Para processar pagamentos e transações financeiras em tempo real, como em bolsas de valores, bancos e seguros.
- Para rastrear e monitorar carros, caminhões, frotas e remessas em tempo real, como na logística e na indústria automotiva.
- Para capturar e analisar continuamente os dados do sensor de dispositivos IoT ou outros equipamentos, como fábricas e parques eólicos.
- Para coletar e reagir imediatamente às interações e pedidos do cliente, como no varejo, no setor de hotéis e viagens e em aplicativos móveis.
- Para monitorar pacientes em cuidados hospitalares e prever mudanças nas condições para garantir o tratamento oportuno em emergências.
- Conectar, armazenar e disponibilizar dados produzidos por diferentes divisões de uma empresa.



# Casos de Uso

- **Spotify**

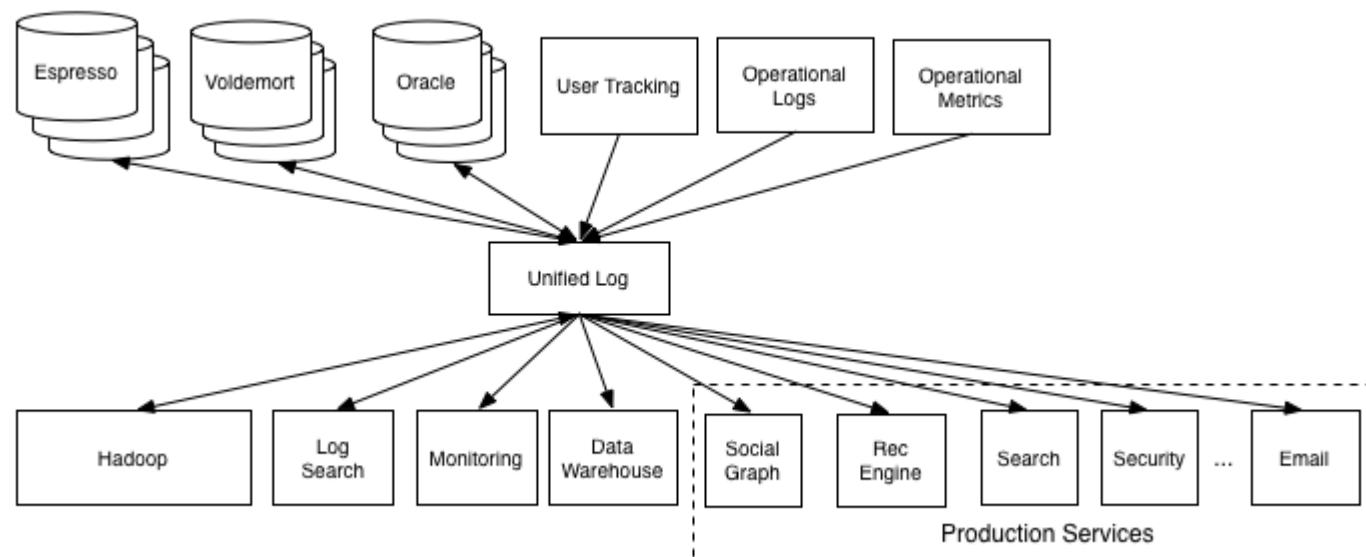
- Utilização do Kafka como um componente central para o sistema de entrega de log. Nosso objetivo é enviar todos os dados produzidos em todos os nossos hosts em nosso cluster Hadoop para processamento posterior. Ao adotar o Kafka como parte de nosso pipeline, conseguimos reduzir o tempo médio necessário para transferir logs **de 4 horas para 10 segundos**.

- **The New York Times**

- Usa o Apache Kafka e a API Kafka Streams para armazenar e distribuir, em tempo real, o conteúdo publicado para os vários aplicativos e sistemas que o tornam disponível aos leitores.

# Implementação do Kafka no Linked-In

- <https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying>



# Livros

- <http://kafka.apache.org/books-and-papers>

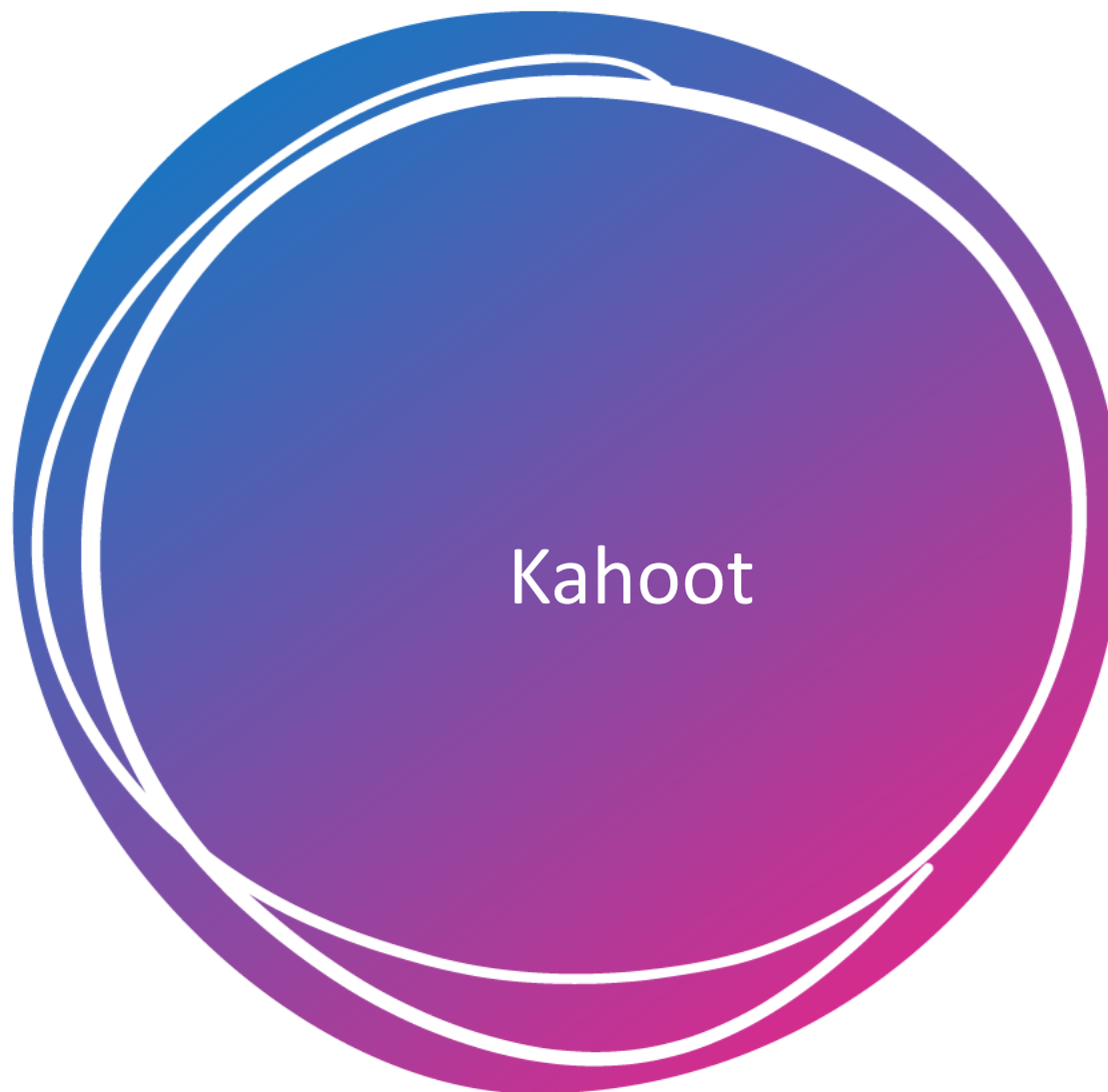
## Books

The following books cover Apache Kafka and/or the subject of event streaming in general.



# Cursos

- <https://www.alura.com.br/formacao-kafka>
- <https://www.linkedin.com/learning/learn-apache-kafka-for-beginners/intro-to-apache-kafka?autoAdvance=true&autoSkip=false&autoplay=true&resume=true&u=89888106>



# Instalação

- Para essa etapa, utilizaremos o Kafka + Zookeeper e Docker + Docker Compose
- git clone <https://github.com/confluentinc/cp-docker-images>
- Navegue até na pasta “cp-docker-images/examples/kafka-single-node”
  - Esta pasta conterá o arquivo “docker-compose.yml”
- Executar comando **docker-compose up -d**
- Baixar o Offset Explorer <https://www.kafkatool.com/download.html>



show me your code;



# Exercício / Homework

- Criar um novo projeto com o spring inicializr com o nome de “produtor-consumidor”
- Colocar o projeto em “modulo 4.3”
- Desenvolver um produtor que produza mensagens simples
- O nome do tópico deve ser “meu-primeiro-topico”
- Criar um serviço (api) para receber essas mensagens e inserir no tópico
- Desenvolver um consumidor (pode ser no mesmo projeto) que consuma as mensagens do tópico e imprima no console