

IMERSÃO KAFKA

ANATOMIA DO APACHE KAFKA





Victor Osório

Twitter: @vepo

Mastodon: mastodon.social/@vepo

LinkedIn: linkedin.com/in/victorosorio/

- Engenheiro de Computação pela UNICAMP
- MBA em Arquitetura de Software na FIAP
- Mestrando em Sistemas Distribuídos na UTFPR
- Arquiteto e desenvolvedor Java/Apache Kafka
- + 18 anos com Java
- + 5 anos com Apache Kafka
- + 3 anos na Amdocs
- Autor: Roadmap back-end (Casa do Código)
 - <https://www.casadocodigo.com.br/products/livro-roadmap-backend>



AGENDA



Decisões
Arquiteturais



Tutoriais e
Dicas



Cluster Kafka



Elementos



Operação



DECISÕES ARQUITETURAIS

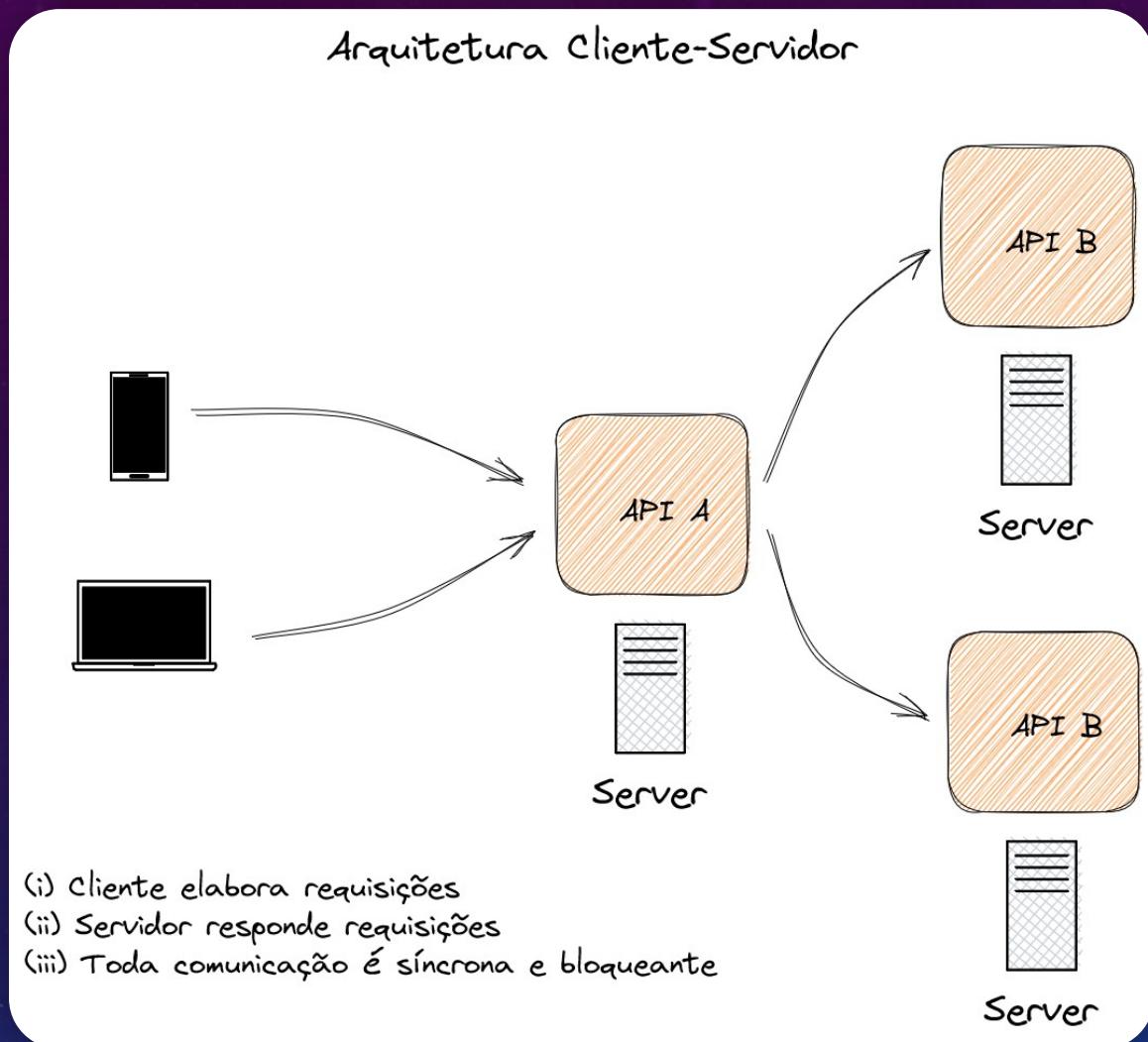
PORQUE APACHE KAFKA? COMO APACHE KAFKA É CONSTRUIDO?

PORQUE KAFKA?

- É Kafka e não Kafta! O nome da comida é Kafta
- Primeiro, um pouco de literatura!
 - Franz Kafka foi um escritor Tcheco (Império Austro-húngaro) boêmio que é muito conhecido por seu romance A Metamorfose.
 - No romance um sujeito de medíocre de classe média acorda como uma barata (inseto desprezível em alemão).
 - O que ele queria dizer é que na sociedade moderna nós podemos se tornar descartáveis do dia para a noite.
- Porque esse nome?
 - É o autor preferido do criado do Apache Kafka Jay Kreps (hoje CEO na Confluent Inc.)
 - O Broker é orientado para escrita, por isso foi escolhido o nome do escritor. Só isso.



COMO É UMA (TÍPICA) ARQUITETURA DE MICROSERVIÇOS?



- APIs HTTP encadeadas
- Escalabilidade
 - Decomposição funcional (Eixo Y)
 - Duplicação (Eixo X)
- Replicação do modelo Cliente-Servidor

LIMITAÇÕES DO MODELO CLIENTE-SERVIDOR

Alto Acoplamento

- API A tem que conhecer B e C
- Disponibilidade da API A é limitada pela B e C

Alta Latência

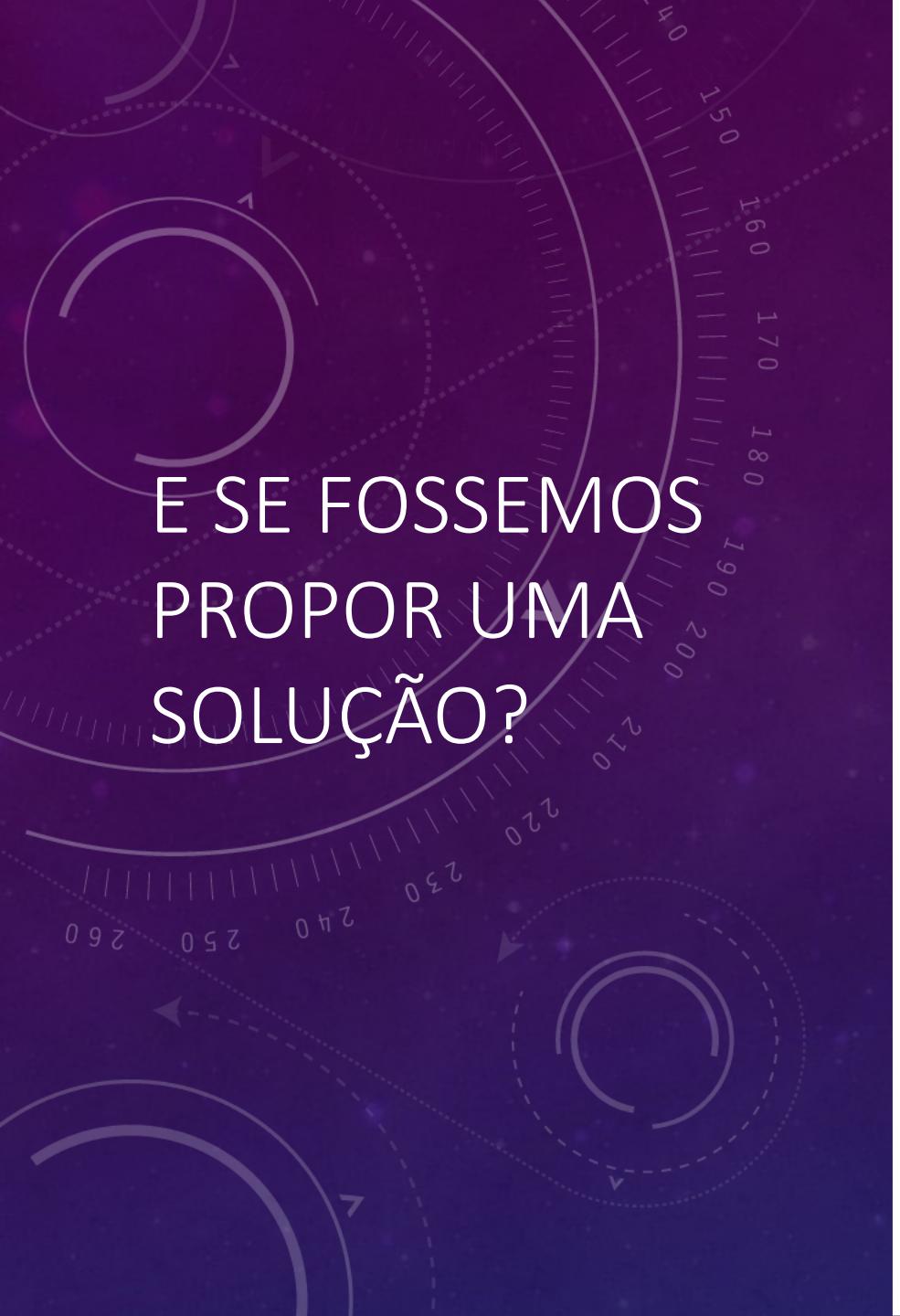
- API A deve esperar que B e C respondam para poder responder suas requisições

Complexidade

- O alto acoplamento traz mais complexidade ao processo, em um mesmo fluxo são encapsulados outros fluxos.

Escalabilidade

- Dificuldade de se escalar por particionamento, apenas por replicação



E SE FOSSEMOS
PROPOR UMA
SOLUÇÃO?

Rápida

Escalável

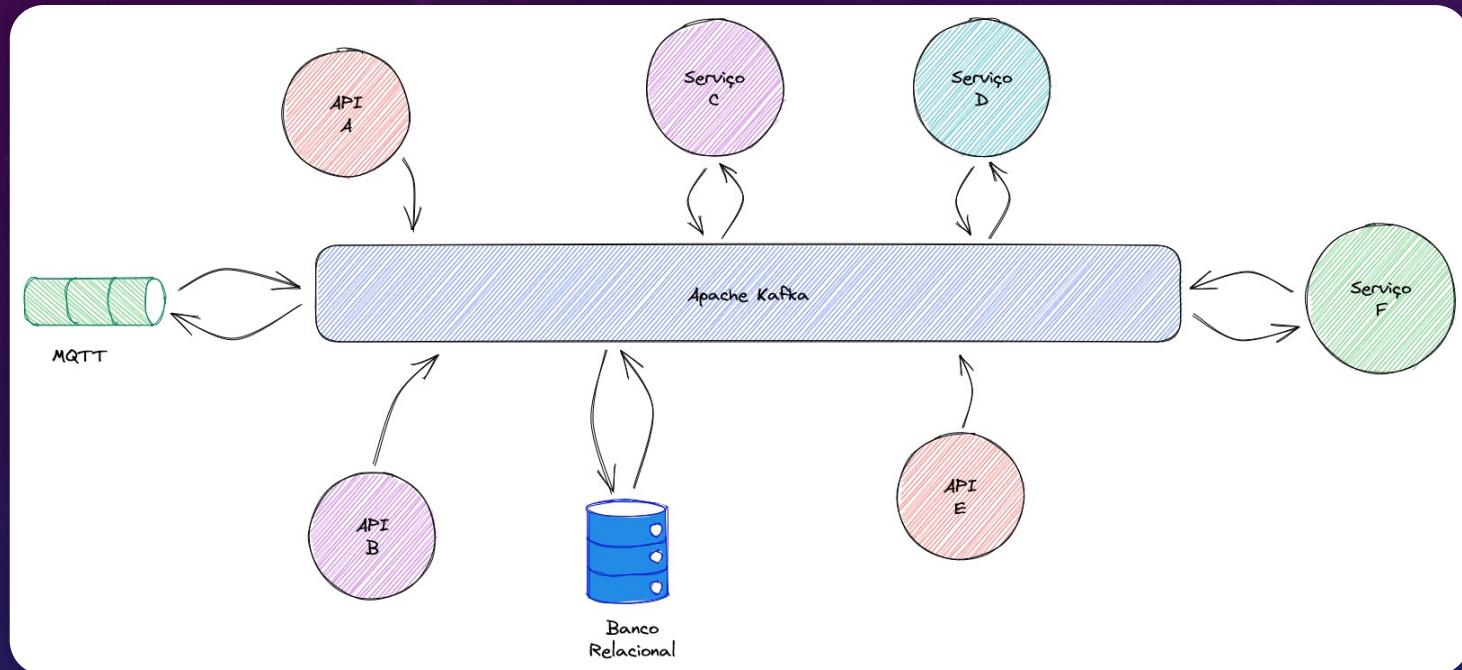
Tolerante a
falhas

Adaptável

Orientada a
Mensagens

Baixo
Acoplamento

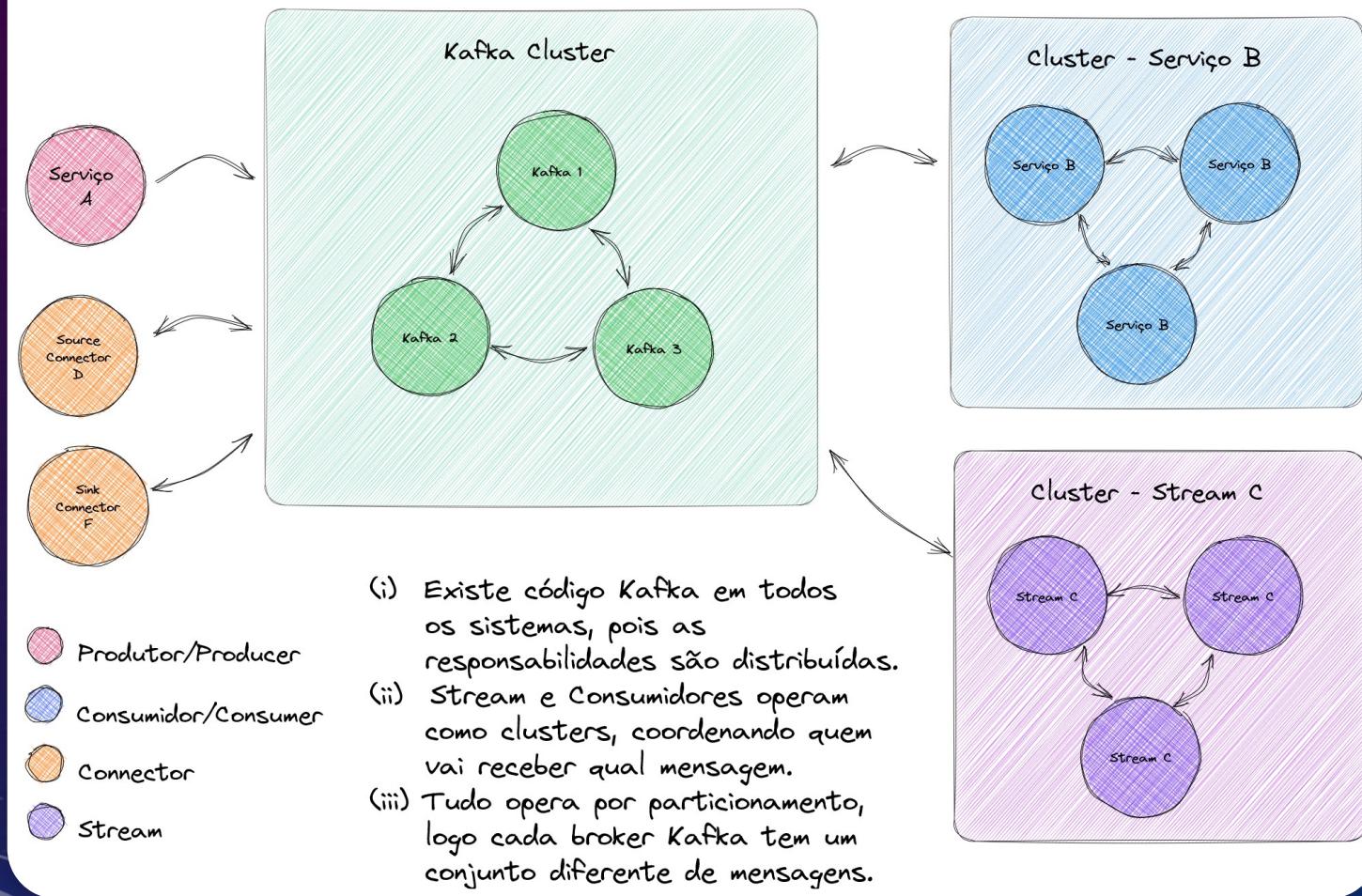
KAFKA COMO BARRAMENTO DE MENSAGENS



- Orientado a Mensagens
- Baixo acoplamento

Fonte: https://excalidraw.com/#json=EgtGhveZIz-m0CEIGGaCd,x2YEt7JI_WlwAt0j51NQpw

Ambiente Kafka



AMBIENTE APACHE KAFKA

- Diversos Clusters distribuídos
- Possibilidade de particionamento e replicação
- Distribuição de responsabilidades
- Processamento em tempo real

Fonte: https://excalidraw.com/#json=bIJx4NOPq-T6FtyWH70HE,ivQGTIkFe5O_SrltSKv0_g

TUTORIAIS E DICAS

COMO ACOMPANHAR ESSE CURSO?



GUIA PRÁTICO

- Esse material está disponível no GitHub
 - <https://github.com/vepo/imersao-kafka>



DICAS



Faça perguntas



Tome notas



Clone o repositório e rode os
exemplos

CLUSTER KAFKA

O QUE CONSISTE O CLUSTER KAFKA



COMO CONFIGURAR UM BROKER

- # 1 Faça o download e siga os passos em <https://kafka.apache.org/quickstart>

STEP 1: GET KAFKA

[Download](#) the latest Kafka release and extract it:

```
1 | $ tar -xzf kafka_2.13-3.4.0.tgz  
2 | $ cd kafka_2.13-3.4.0
```

(i) Baixe o arquivo em "Download" e salve em algum diretório da sua máquina.

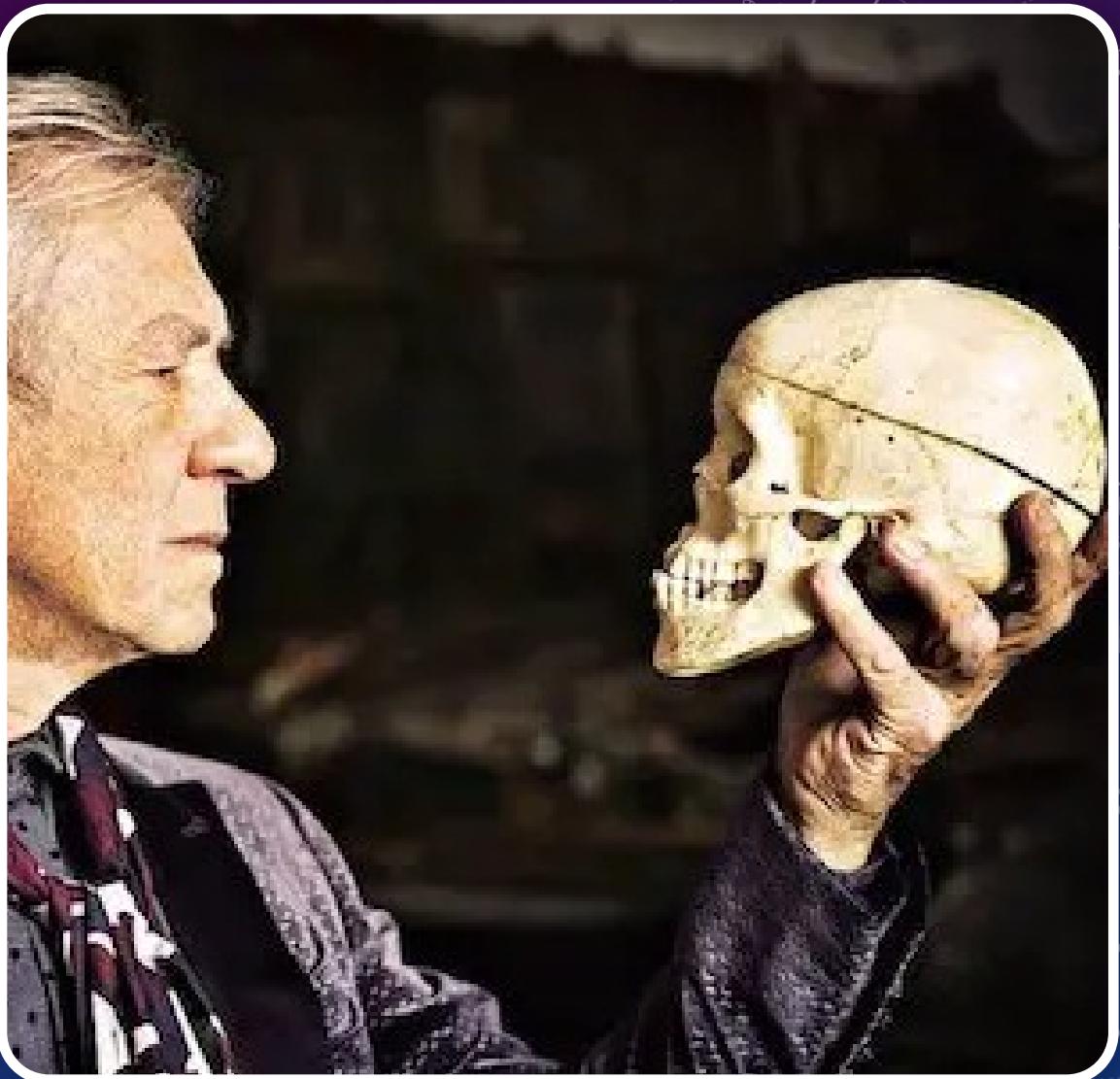
(ii) Extraia ele em um diretório

-> No Windows uso C:\opt\kafka

-> No Linux uso /opt/kafka

ZOOKEEPER OR NOT ZOOKEEPER

- Zookeeper
 - Legado
 - Controle de offsets
 - Leader Election
 - Será removido na 4.0
- Apenas Kafka
 - Disponível a partir da 3.3
 - Kraft



CONHEÇA AS PROPRIEDADES

- * auto.create.topics.enable
- * delete.topic.enable
- * background.threads
- * log.retention.*
- * log.roll.*
- * log.segment.*
- * min.insync.replicas
- * num.io.threads
- * num.network.threads
- * num.recovery.threads.per.data.dir
- * num.partitions
- * default.replication.factor
- * offsets.retention

3.1 Broker Configs

The essential configurations are the following:

- broker.id
- log.dirs
- zookeeper.connect

Topic-level configurations and defaults are discussed in more detail [below](#).

ELEMENTOS

O QUE COMPÕE UM SISTEMA QUE USA APACHE KAFKA?



4 ELEMENTOS BÁSICOS



Produtor

Apenas envia mensagens



Consumidor

Apenas consome mensagens
Controle de offset
Formação de Cluster (group.id)



Tópico

Canal de Comunicação
Particionamento
Replicação
Log/Offset



Stream

Processamento de mensagens em tempo real
Consome e envia
Controle de offset
Formação de Cluster (group.id)



Connect

Connect é um tipo de servidor de aplicação
Connector é o processo que conecta o Cluster Kafka a uma fonte de dados
Sources e Sinks

DELEGAR PARA CONQUISTAR

PRINCÍPIO BÁSICO PARA ESCALABILIDADE



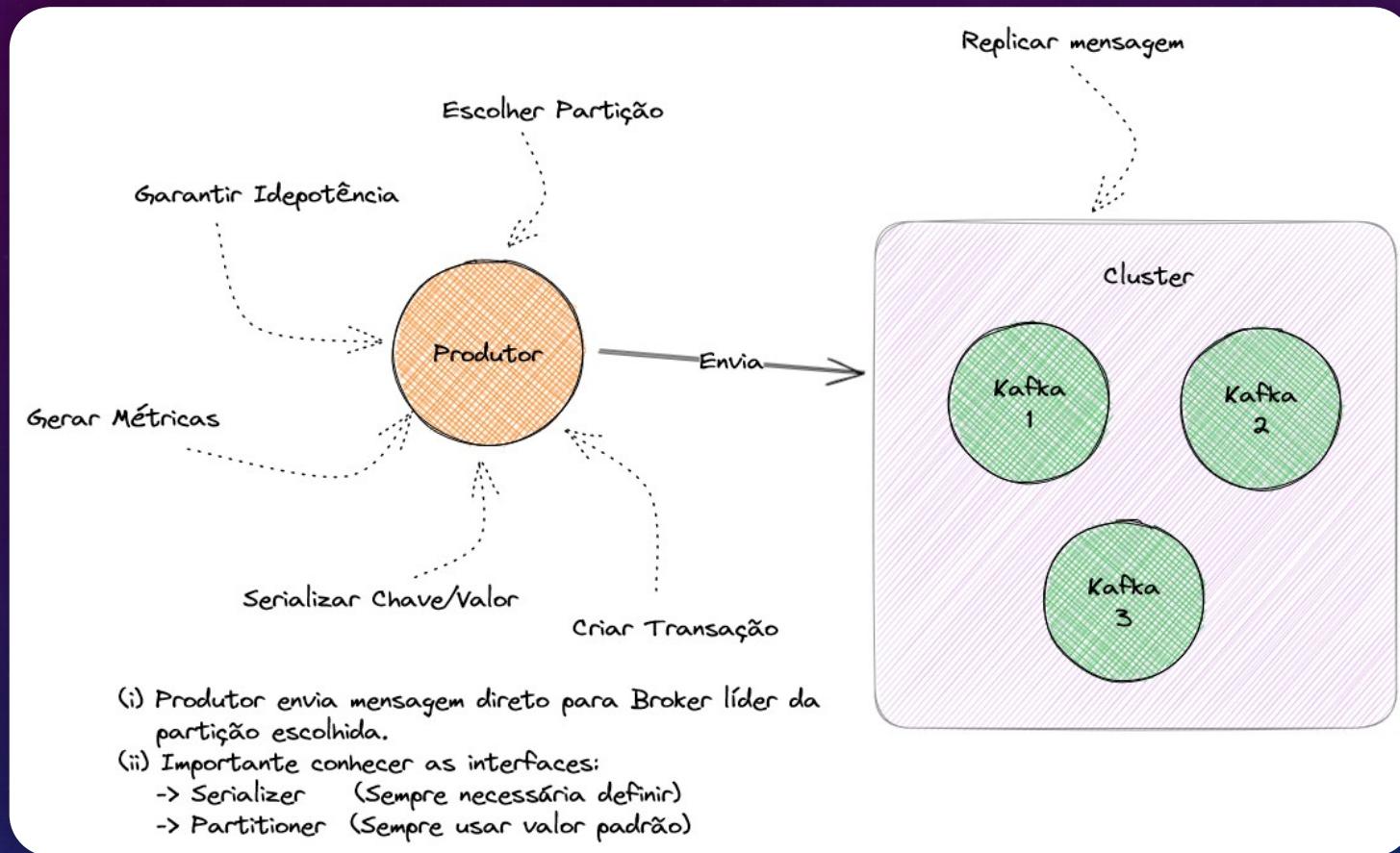
- Kafka atinge um elevado *throughput* porque delega responsabilidades aos clientes
 - Ao broker cabe apenas a responsabilidade de ler/enviar mensagens

PRODUTOR

COMO IMPLEMENTAR E RESPONSABILIDADES

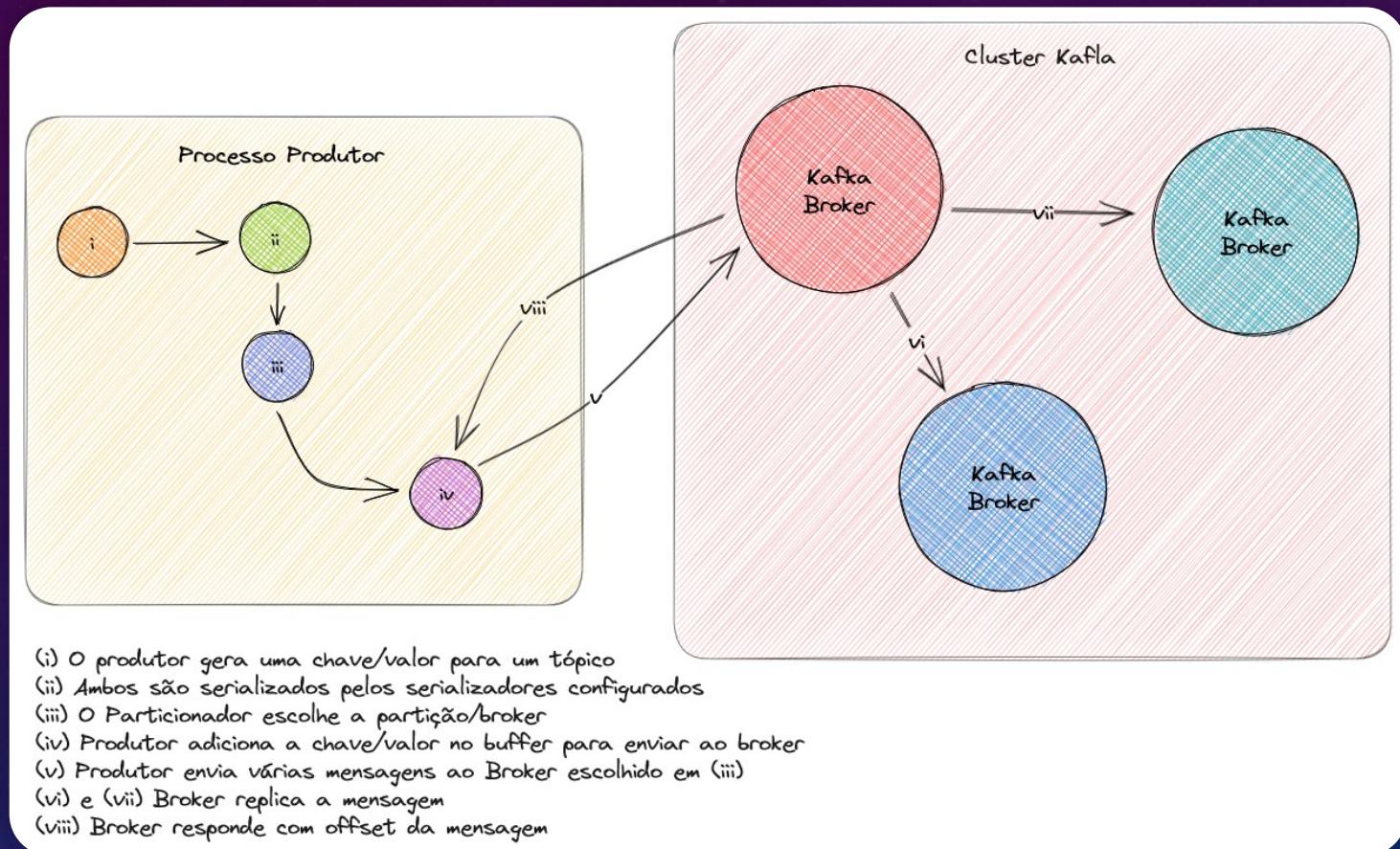


RESPONSABILIDADES



Fonte: https://excalidraw.com/#json=ybjWoVz1mh_xBfQ0sQ7w,8QzzmXstRNAdFp07-ACcPg

ENVIO DA CHAVE/VALOR



Fonte: https://excalidraw.com/#json=BHFfh3_IknMEaL95xDQ3X,RHy3KEW236CyO8y_VXkbew

- Partição/Broker é escolhido pelo cliente
- Kafka deve receber a mensagem e replica
- Offset definido pelo broker

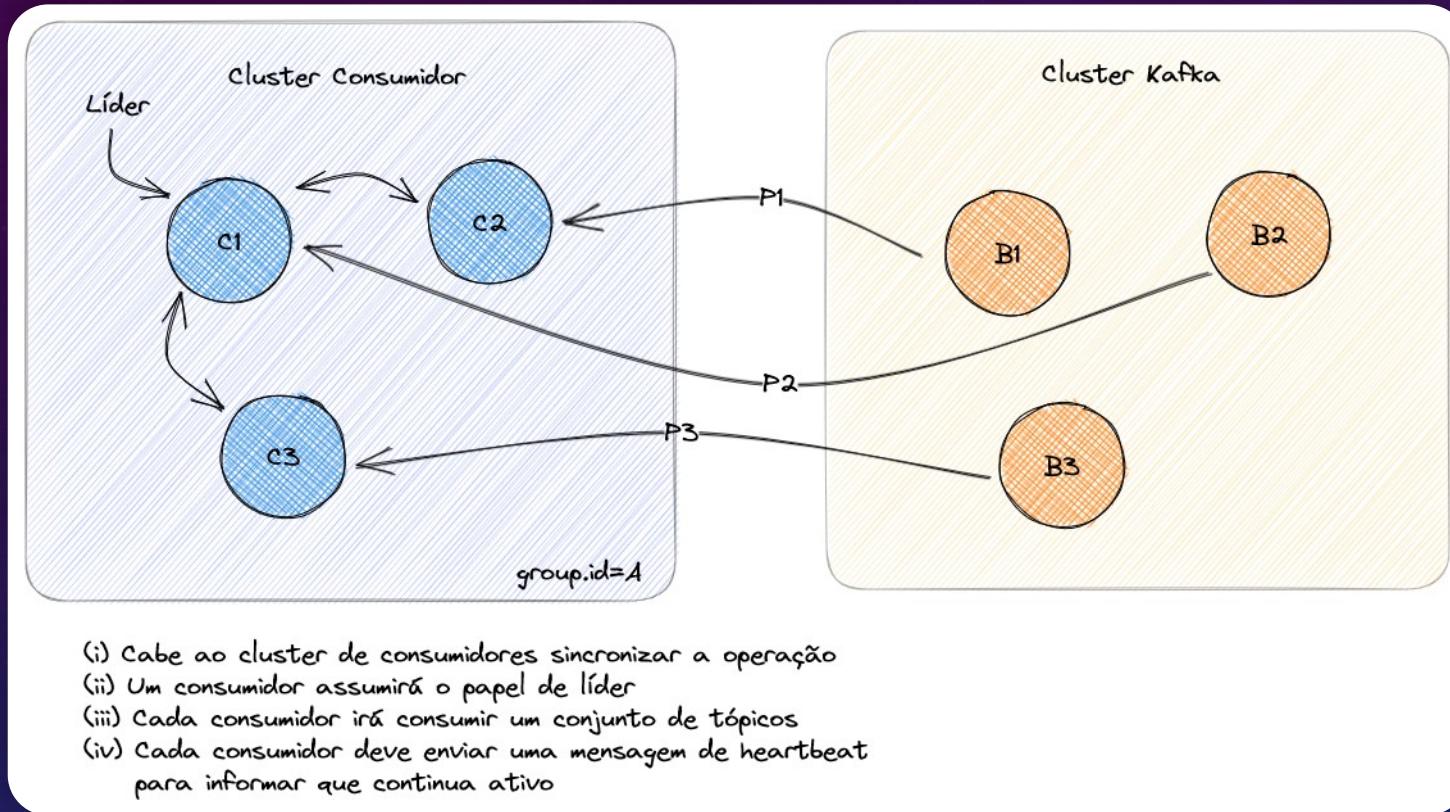
CONSUMIDOR

COMO IMPLEMENTAR E RESPONSABILIDADES



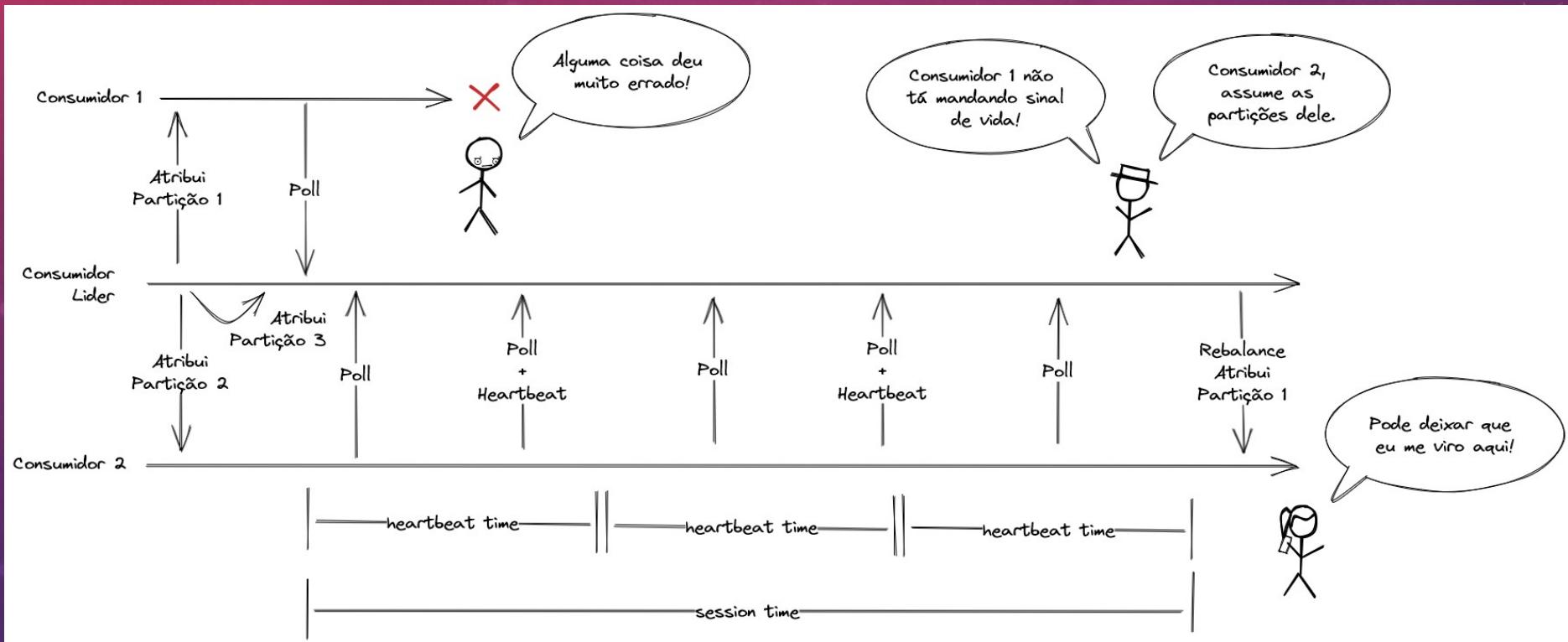
RESPONSABILIDADES

- Coordenar cluster de consumidor
- Consumir um conjunto de partições
- Distribuir carga entre consumidores ativos



Fonte: https://excalidraw.com/#json=fLS5sznS8_hlcDB6hzPhIuaNi0KcSzidbQlxmPA

HEARTBEATING/REBALANCEAMENTO



Fonte: https://excalidraw.com/#json=gZx0kObUVMc-lG6stv7FT,_KJ_e8BtYMvI2K8uuj1kZQ

TÓPICO

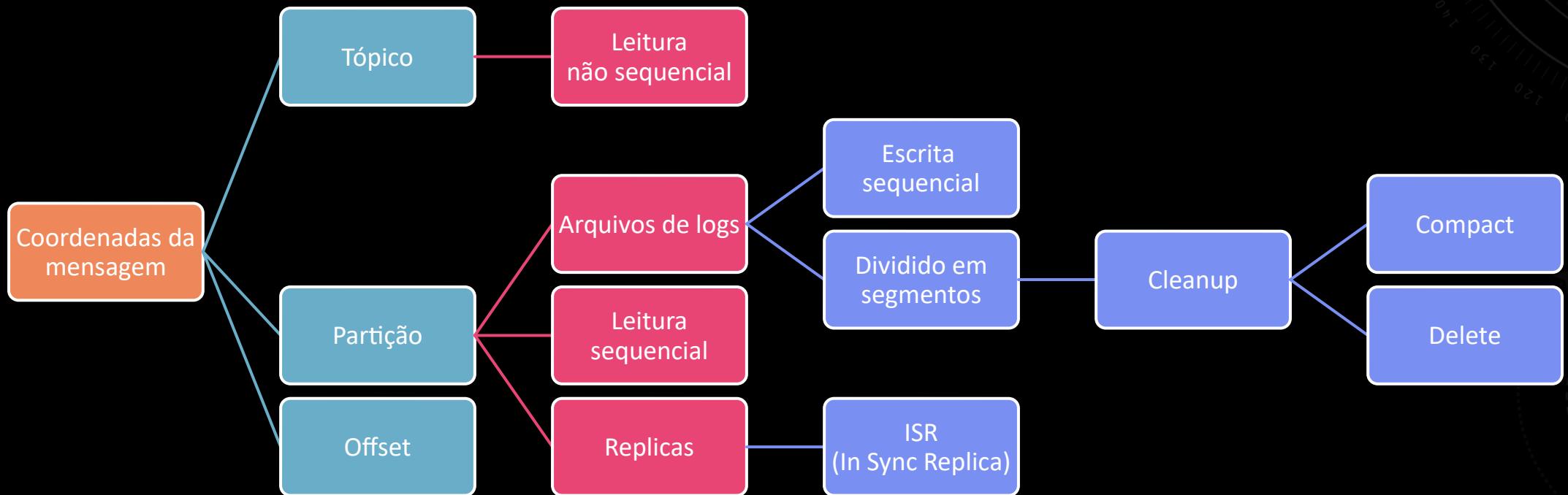
ANATOMIA



Anatomia de um Tópico



TÓPICO



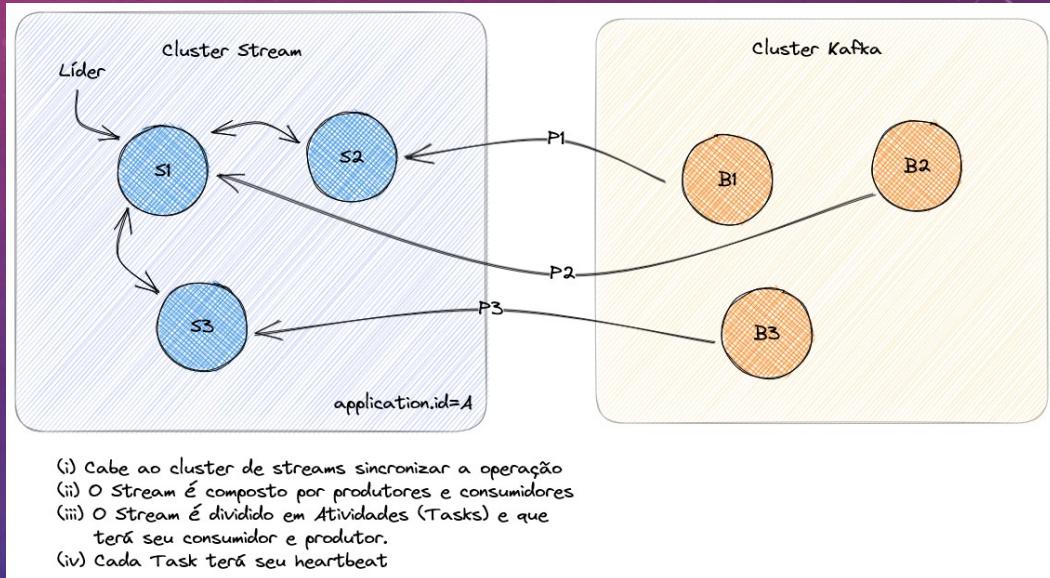
STREAM

COMO IMPLEMENTAR E RESPONSABILIDADES



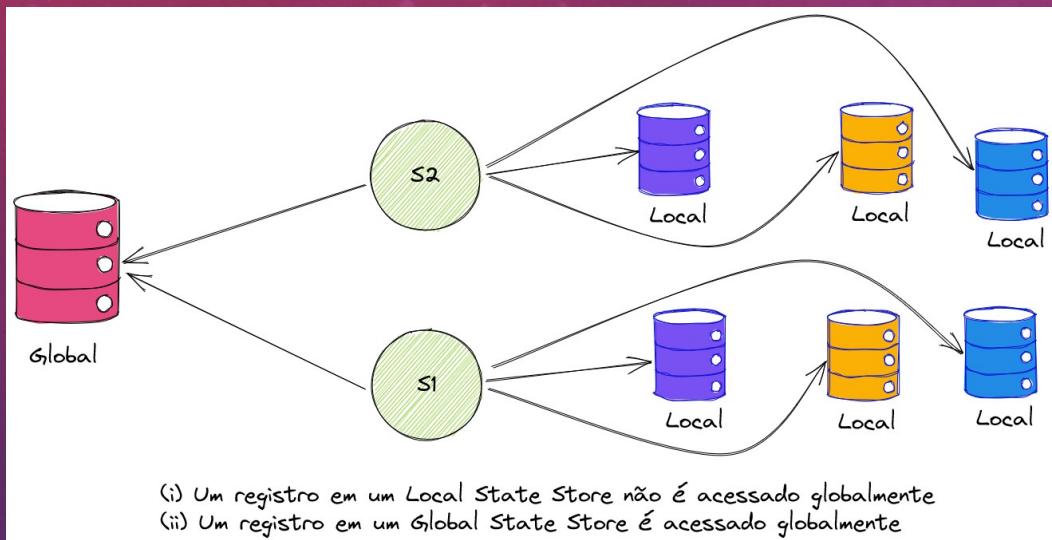
RESPONSABILIDADES

- Um Stream tem responsabilidade parecida com um Consumidor
- Mas o *poll* será feito pelo Stream



Fonte: https://excalidraw.com/#json=wpSMzndwmvwMrNPM7N-4YTQ_qu_Tp0NrtQKEVs9rXdQ

STATE STORES



- Tópico

Global



- Partição

Local



Fonte: <https://excalidraw.com/#json=4GxJLNUvW2uIriqmC8n1v,Fn34avDW5E4T7tydu-ys0w>

CONNECT

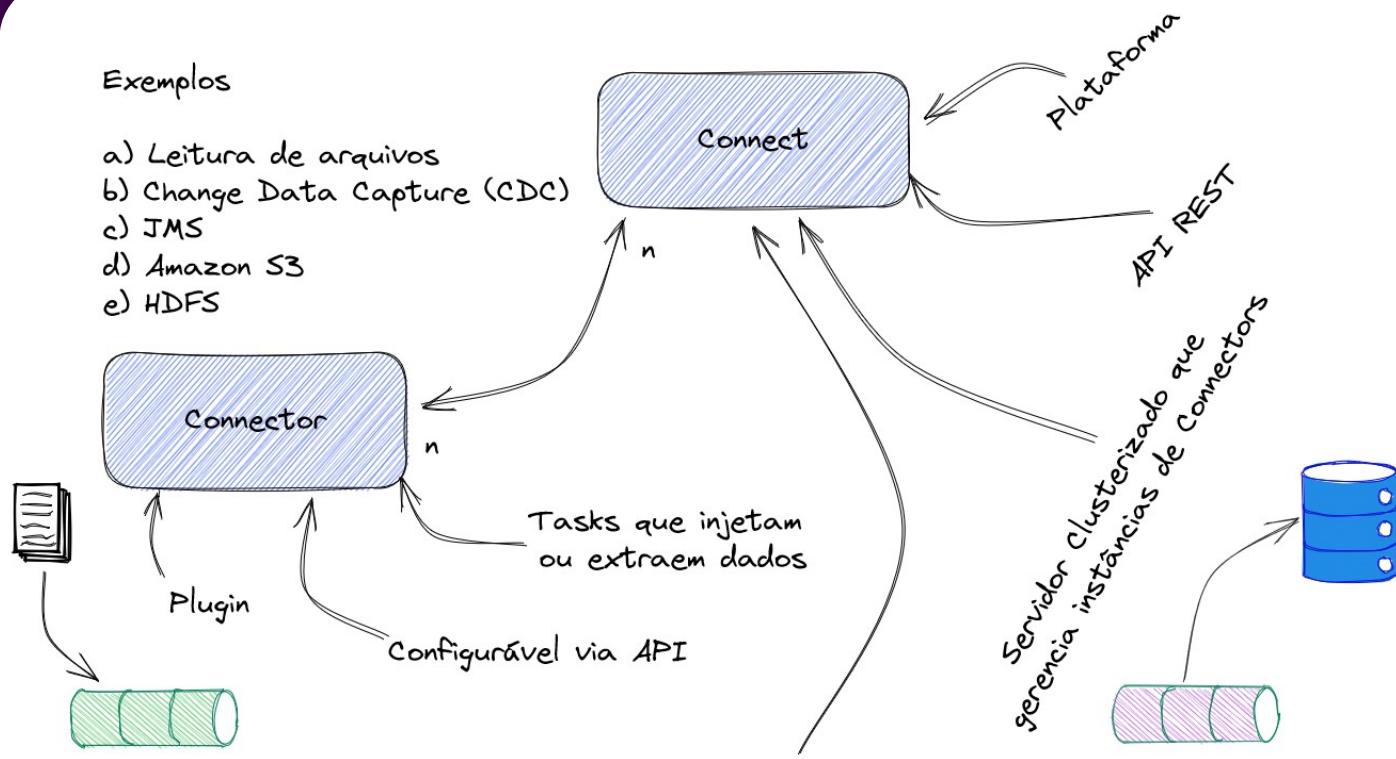
RESPONSABILIDADES E CONFIGURAÇÃO



RESPONSABILIDADES E CONFIGURAÇÃO

Exemplos

- a) Leitura de arquivos
- b) Change Data Capture (CDC)
- c) JMS
- d) Amazon S3
- e) HDFS



```
POST http://kafka-connect:8083/connectors
{
  "config" : {
    "name" : "RedisSinkConnector1",
    "connector.class" : "com.github.jcugenborder.kafka.connect.redis.RedisSinkConnector",
    "tasks.max" : "1",
    "topics" : "topic",
    "redis.hosts": "some-host"
  }
}
```

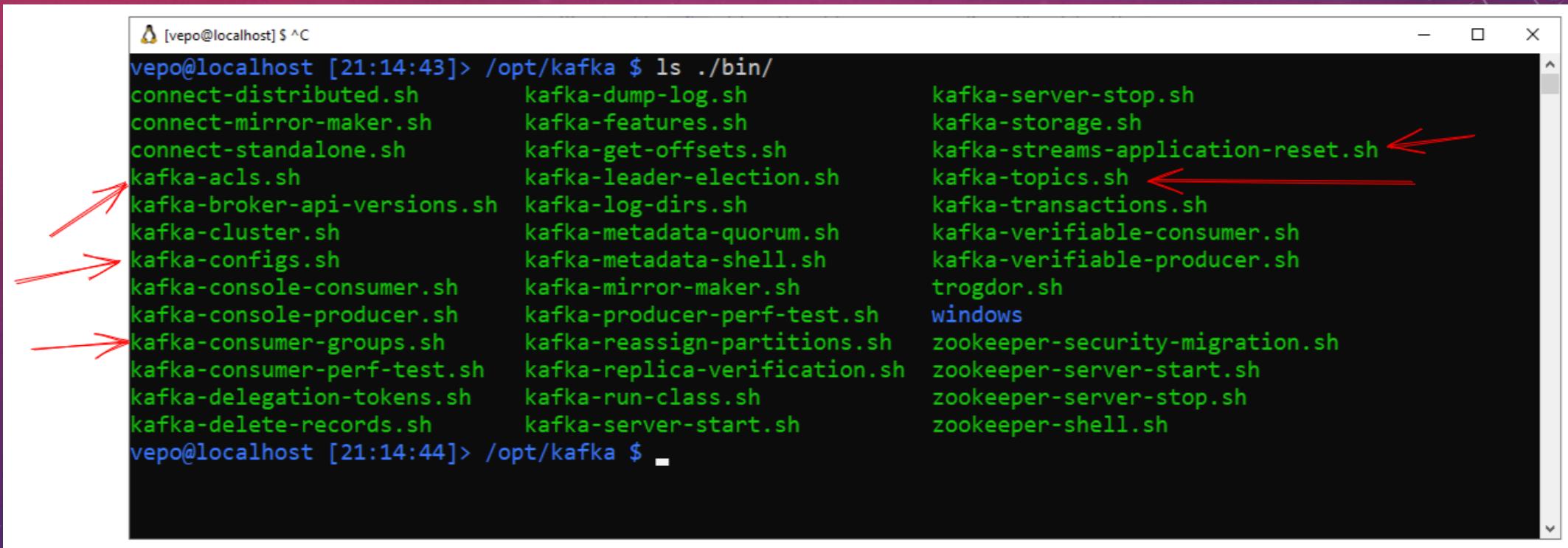
Fonte: <https://excalidraw.com/#json=Qy1wwGPaeQMSH-IMBLqCB,o9LnihBKsasvmMhQTF1Mow>

OPERAÇÃO

COMANDOS ÚTEIS



COMANDOS IMPORTANTES



```
vepo@localhost [21:14:43]> /opt/kafka $ ls ./bin/
connect-distributed.sh          kafka-dump-log.sh
connect-mirror-maker.sh         kafka-features.sh
connect-standalone.sh           kafka-get-offsets.sh
kafka-acls.sh                  kafka-leader-election.sh
kafka-broker-api-versions.sh    kafka-log-dir.sh
kafka-cluster.sh                kafka-metadata-quorum.sh
kafka-configs.sh                kafka-metadata-shell.sh
kafka-console-consumer.sh       kafka-mirror-maker.sh
kafka-console-producer.sh       kafka-producer-perf-test.sh
kafka-consumer-groups.sh        kafka-reassign-partitions.sh
kafka-consumer-perf-test.sh     kafka-replica-verification.sh
kafka-delegation-tokens.sh      kafka-run-class.sh
kafka-delete-records.sh         kafka-server-start.sh
vepo@localhost [21:14:44]> /opt/kafka $ _
```

```
kafka-server-stop.sh
kafka-storage.sh
kafka-streams-application-reset.sh
kafka-topics.sh
kafka-transactions.sh
kafka-verifiable-consumer.sh
kafka-verifiable-producer.sh
trgdor.sh
windows
zookeeper-security-migration.sh
zookeeper-server-start.sh
zookeeper-server-stop.sh
zookeeper-shell.sh
```

Fonte: <https://excalidraw.com/#json=ayCkKYK8ukwT-GKrbibv,7xUqy8Zlpu07imAQFMS8Fg>

DIRETÓRIOS IMPORTANTES

- (i) Todas as informações são salvas em disco
- (ii) Um diretório para cada partição

Valor de log.dirs nas configurações do broker

```
vypo@localhost [21:22:08]> /opt/kafka $ ls /tmp/kraft-combined-logs
cluster_metadata-0    consumer_offsets-21  consumer_offsets-35  consumer_offsets-49
consumer_offsets-0    consumer_offsets-22  consumer_offsets-36  consumer_offsets-5
consumer_offsets-1    consumer_offsets-23  consumer_offsets-37  consumer_offsets-6
consumer_offsets-10   consumer_offsets-24  consumer_offsets-38  consumer_offsets-7
consumer_offsets-11   consumer_offsets-25  consumer_offsets-39  consumer_offsets-8
consumer_offsets-12   consumer_offsets-26  consumer_offsets-4  consumer_offsets-9
consumer_offsets-13   consumer_offsets-27  consumer_offsets-40  bootstrap.checkpoint
consumer_offsets-14   consumer_offsets-28  consumer_offsets-41  cleaner-offset-checkpoint
consumer_offsets-15   consumer_offsets-29  consumer_offsets-42  log-start-offset-checkpoint
consumer_offsets-16   consumer_offsets-3  consumer_offsets-43  meta.properties
consumer_offsets-17   consumer_offsets-30  consumer_offsets-44  recovery-point-offset-checkpoint
consumer_offsets-18   consumer_offsets-31  consumer_offsets-45  replication-offset-checkpoint
consumer_offsets-19   consumer_offsets-32  consumer_offsets-46  weather-0
consumer_offsets-2    consumer_offsets-33  consumer_offsets-47  weather-sanity-weather-memory-changelog-0
consumer_offsets-20   consumer_offsets-34  consumer_offsets-48  weather-stable-0
vypo@localhost [21:22:09]> /opt/kafka $
```

Fonte: https://excalidraw.com/#json=L8Y0fazIM_d91fdG66eZi,8NLfcVZuuj8IYEVmQv6WA

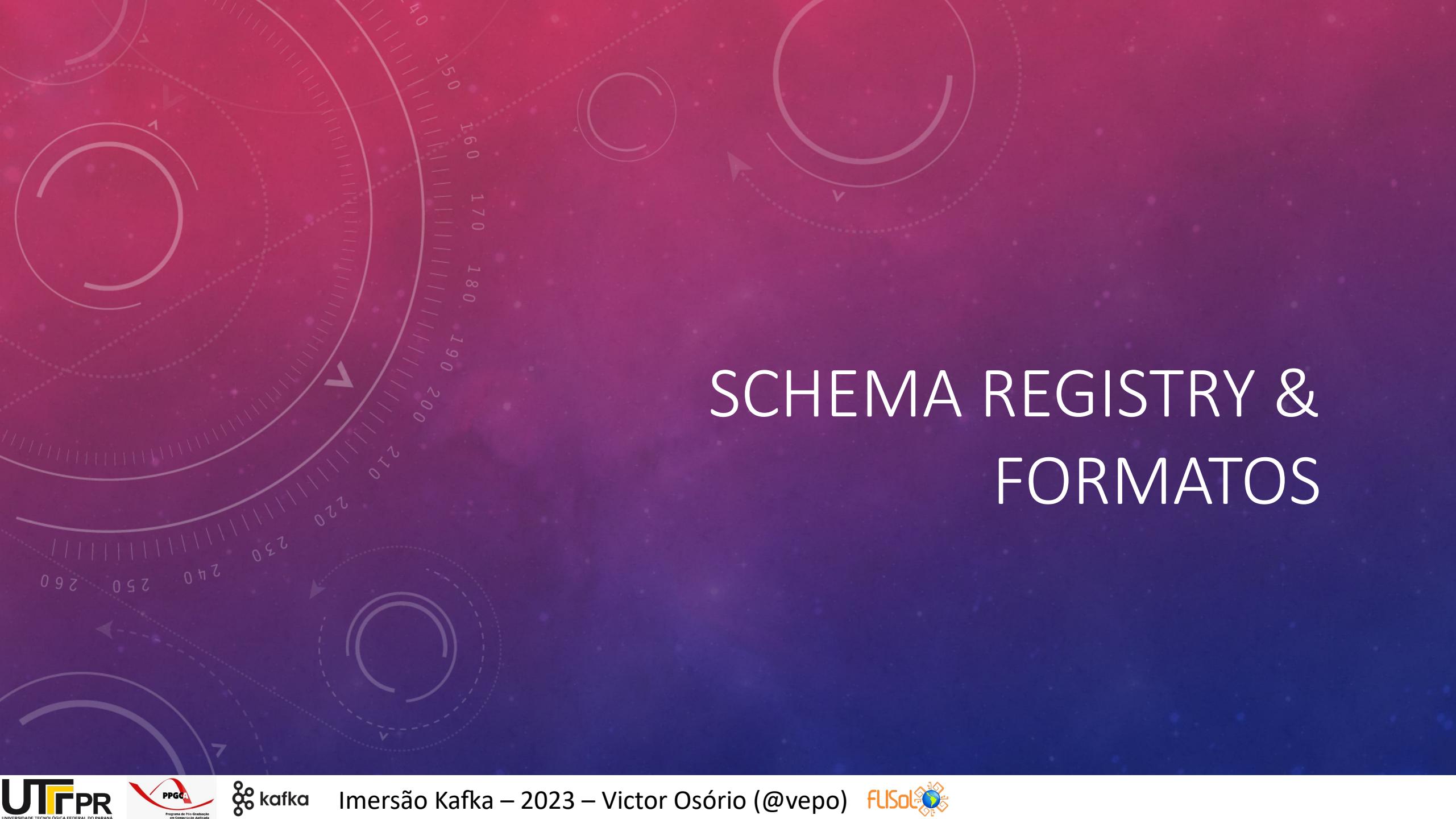
DIRETÓRIOS IMPORTANTES

```
vepo@localhost $ ^C
vepo@localhost [21:26:44]> /opt/kafka $ ls /tmp/kraft-combined-logs/weather-0/
00000000000000000000.index 00000000000000000000.timeindex 00000000000000003259.snapshot partition.metadata
00000000000000000000.log    00000000000000002263.snapshot leader-epoch-checkpoint
vepo@localhost [21:26:52]> /opt/kafka $ -
```

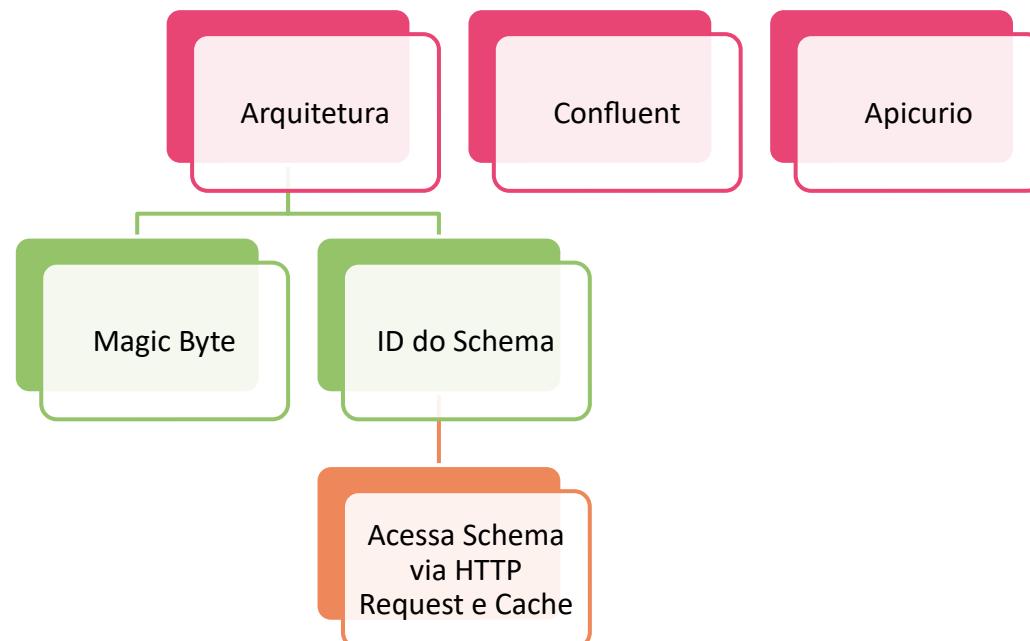
(i) Contém todas as informações da partição
(ii) Cada arquivo *.log é um arquivo de segmento

Fonte: <https://excalidraw.com/#json=MmOBDJ7k11aXxeoH1BUdQ,4pWmEfnsKoZ8MhUREjKFVA>

SCHEMA REGISTRY & FORMATOS



SCHEMA REGISTRY



FORMATOS

JSON

AVRO

Protobuf

DESAFIO!!!!!

- Configurar um cluster com 3 brokers na mesma máquina
- Configurar o consumidor
- Configurar o produtor



PROJETOS OPEN SOURCE INTERESSANTES



PROJETOS OPEN SOURCE INTERESSANTES

Strimzi

- Kafka como um Recurso Kubernetes

AsyncAPI

- Documentação Event-Driven

Kafka

- Dispensa apresentações

Ideias?!?!

- Framework Low-Code para Stream
- UI para gerenciamento do cluster

OBRIGADO!

OSORIO@ALUNOS.UTFPR.EDU.BR

