

# Hands On Kafka

## Preparação do ambiente

- Instale o docker e docker-compose
- Neste documento usei a versão 2.8.0 do Kafka Binaries:
  - <https://kafka.apache.org/downloads.html>
- Github: <https://github.com/mateuspada/demo-kafka-stack>

## Primeiros passos - produtor e consumidor

Inicie um único broker Kafka com docker-compose:

```
docker-compose -f kafka-single.yml up
```

Crie o primeiro tópico:

```
kafka-topics.bat --bootstrap-server localhost:9092 --create --topic first_topic --replication-factor 1 --  
partitions 1
```

Para listar os tópicos criados, use o comando abaixo:

```
kafka-topics.bat --bootstrap-server localhost:9092 --list
```

E se quiser ver mais detalhes do tópico criado:

```
kafka-topics.bat --bootstrap-server localhost:9092 --topic first_topic --describe
```

Para produzir mensagens (o cursor ficará aguardando digitar mensagens, para enviar pressione enter):

```
kafka-console-producer.bat --broker-list localhost:9092 --topic first_topic
```

Para consumir mensagens:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic first_topic
```

Nesse momento nenhuma mensagem será exibida. Uma forma de "resolver" é adicionar `--from-beginning`:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic first_topic --from-beginning
```

**OBS:** Todas as vezes que iniciamos um consumidor com `kafka-console-consumer` um novo grupo de consumidores é criado (caso não usemos o parâmetro `--group`), e o novo grupo inicia no último offset - caso não usemos o parâmetro `--from-beginning` - que faz recuperar as mensagens do início do offset, mas não o altera, veremos mais sobre isso.

Para listar os consumer groups existentes:

```
kafka-consumer-groups.bat --bootstrap-server localhost:9092 --list
```

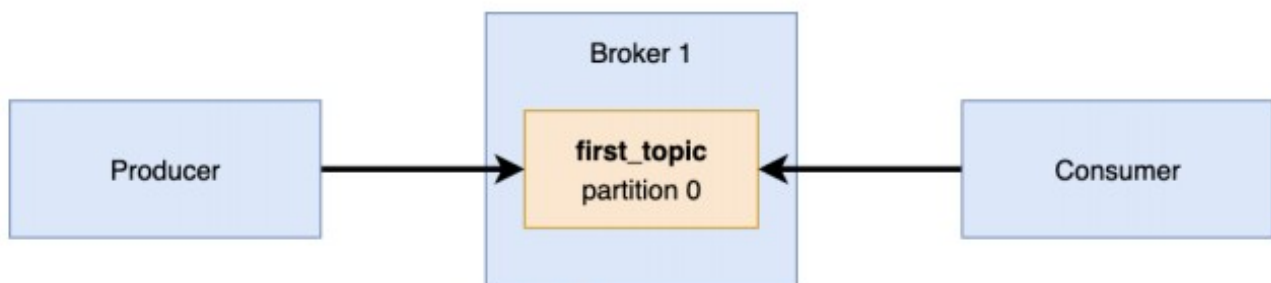


Figura 1 - Primeiros passos

## Grupos de consumidores

Crie um segundo tópico:

```
kafka-topics.bat --bootstrap-server localhost:9092 --create --topic second_topic --replication-factor 1 --partitions 1
```

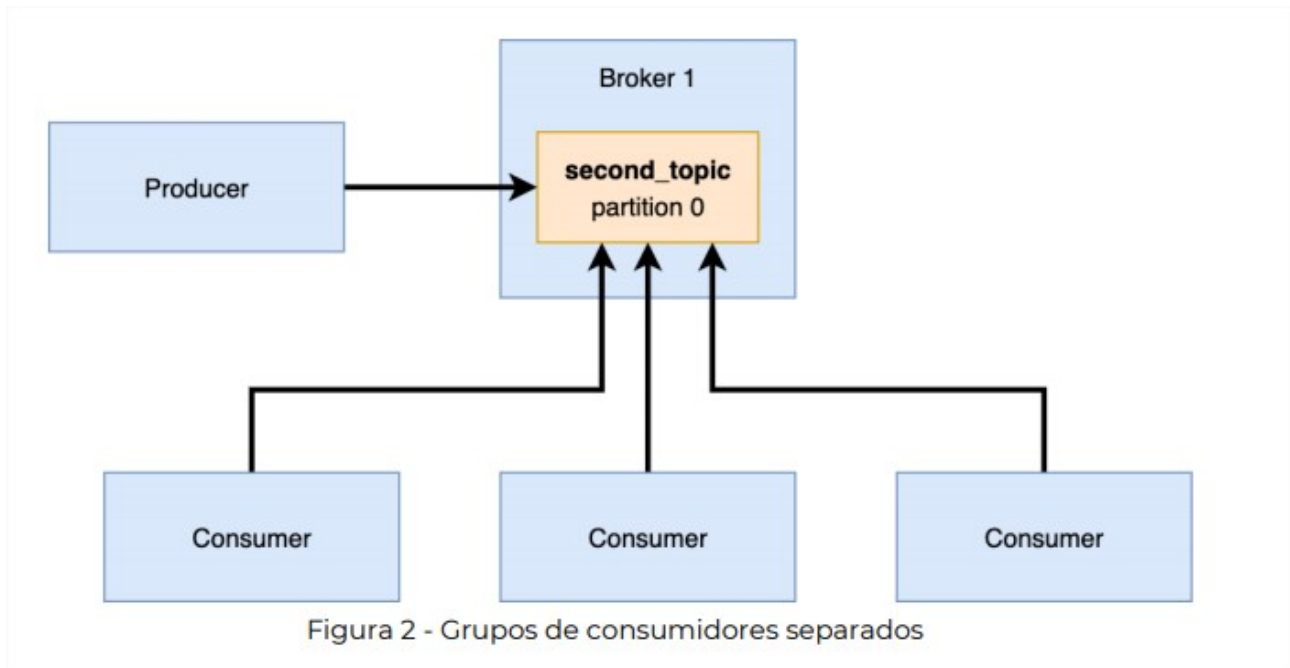
Inicie 3 consumidores em terminais separados:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic second_topic
```

Produza mensagens com conteúdos aleatórios:

```
kafka-console-producer.bat --broker-list localhost:9092 --topic second_topic
```

TODOS os consumidores receberam a mensagem, pois tem consumers groups diferentes, como mostra a imagem abaixo:



Podemos iniciar os consumidores no mesmo grupo de consumidores usando a opção `--group`.

Em 3 terminais inicie os consumidores com o comando abaixo:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic second_topic --group app
```

Agora apenas UM consumidor recebe cada mensagem. A figura 3 ilustra os consumidores sendo parte de um mesmo grupo.

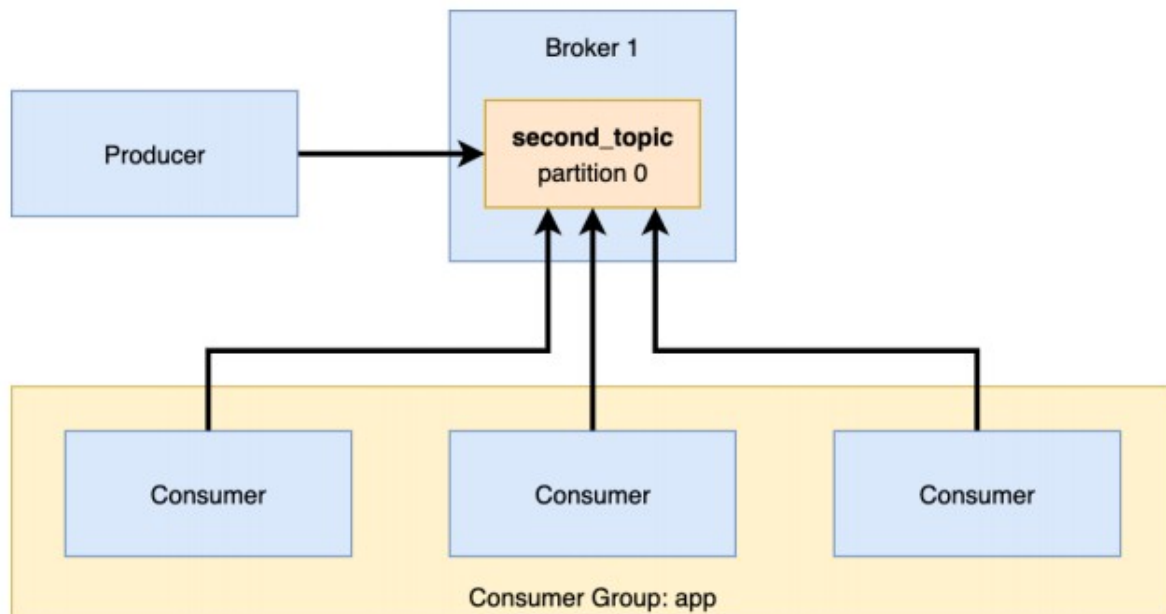


Figura 3 - Os consumidores fazem parte de um grupo

## Aprendendo sobre partições

Crie um novo tópico com 3 partições:

```
kafka-topics.bat --bootstrap-server localhost:9092 --create --topic third_topic --replication-factor 1  
--partitions 3
```

Inicie 3 consumidores em terminais diferentes no mesmo grupo:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic third_topic --group app
```

Produza mensagens para o tópico:

```
kafka-console-producer.bat --broker-list localhost:9092 --topic third_topic
```

Repare que cada consumidor recebe as mensagens como mostra a figura abaixo:

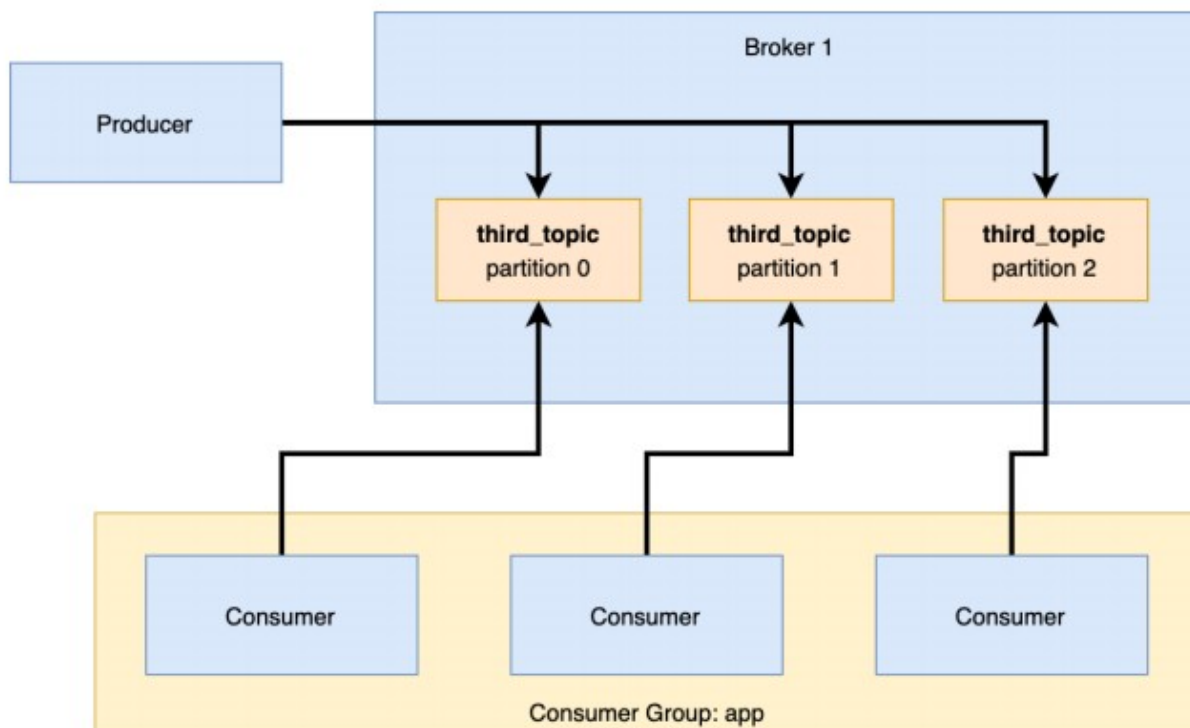


Figura 4 - Exemplo de partições

Caso um dos consumidores pare por algum problema, outro consumidor assume a partição como mostra a figura 5 abaixo:

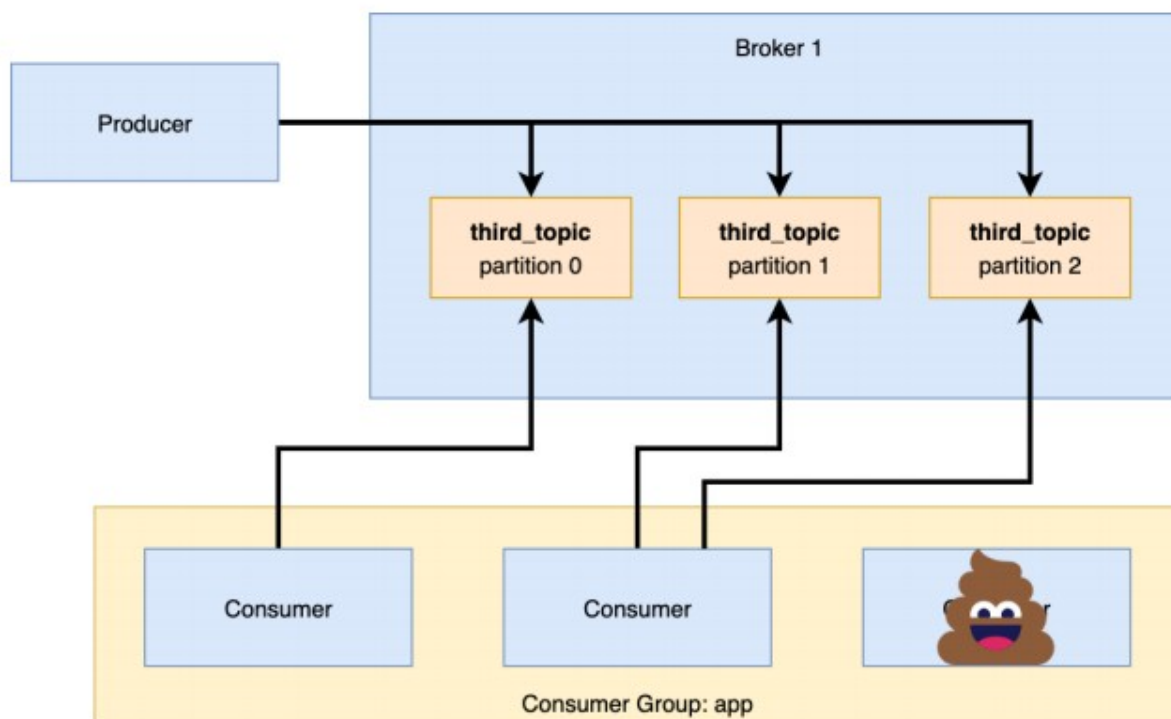


Figura 5 - Um consumidor com 2 partições

## Direcionamento das partições por chave

Crie um novo tópico com 3 partições:

```
kafka-topics.bat --bootstrap-server localhost:9092 --create --topic fourth_topic --replication-factor 1 --partitions 3
```

Em 3 terminais diferentes, inicie os consumidores no mesmo grupo:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic fourth_topic --group app
```

Vamos produzir mensagens com chave usando as propriedades `parse.key` e `key.separator`:

```
kafka-console-producer.bat --broker-list localhost:9092 --topic fourth_topic --property parse.key=true --property key.separator=:
```

A chave é separada por dois pontos da mensagem, como abaixo:

```
a:mensagem1
```

```
b:mensagem2
```

Repare que a mensagem com a mesma chave é sempre enviada para o mesmo consumidor.

Se você parar um dos consumidores, outro assumirá a partição devido ao rebalance.

## Mais sobre offset

Crie o tópico abaixo:

```
kafka-topics.bat --bootstrap-server localhost:9092 --create --topic fifth_topic --replication-factor 1 --partitions 1
```

Crie um novo consumidor:

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic fifth_topic --group app
```

Produza algumas mensagens:

```
kafka-console-producer.bat --broker-list localhost:9092 --topic fifth_topic
```

Pare o consumidor e inicie novamente. Repare que não mostrou as mensagens, mesmo se adicionar o `--from-beginning`, faça o teste.

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic fifth_topic --group app --  
from-beginning
```

Isso se dá por conta do controle de offset do consumer-group. Para visualizar o offset do consumer-group:

```
kafka-consumer-groups.bat --bootstrap-server localhost:9092 --group app --describe
```

Pare o consumidor e volte o offset em 2 posições com o comando abaixo:

```
kafka-consumer-groups.bat --bootstrap-server localhost:9092 --group app --reset-offsets --execute  
--topic fifth_topic --shift-by -2
```

Inicie o consumidor novamente e veja que 2 mensagens serão consumidas.

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic fifth_topic --group app
```

**OBS:** A opção `--from-beginning` é utilizado apenas na criação de um consumer-group novo para dizer que o mesmo iniciará com o offset desde o início do tópico (offset 0), após criado, é indiferente a utilização do mesmo, conforme exemplo acima.

## Sobre as réplicas