

The background features a dark blue gradient with several digital-themed elements. On the left, there is a bar chart with green bars of varying heights. A red line graph is overlaid on the chart, showing an upward trend. Scattered throughout the background are binary digits (0s and 1s) in white and light blue. In the lower right, there is a network diagram with red and blue nodes connected by lines, forming a complex web. The overall aesthetic is futuristic and tech-oriented.

FIAP

# PLATAFORMAS E SERVIÇOS COGNITIVOS

PROF. DR. FERNANDO TIMOTEO FERNANDES

# Agenda

---

- Introdução ao Apache Kafka
- Event Streams



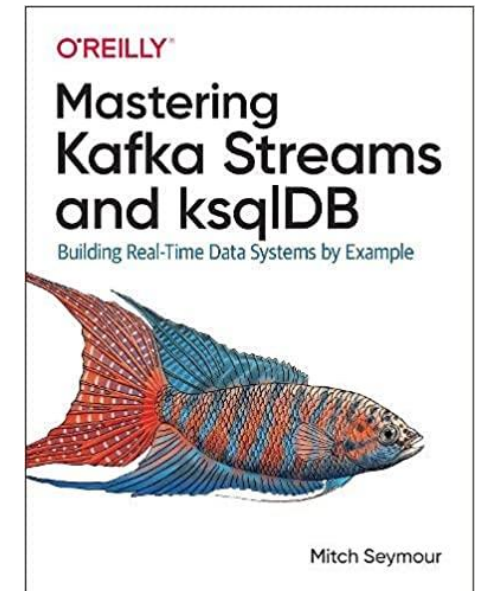
Apacha Kafka

# Kafka - Definição

---

“Apache Kafka (or simply, Kafka) is a streaming platform for ingesting, storing, accessing, and processing streams of data.”

Fonte: Seymour (2021)



# Kafka

---

- Sistema de mensagens distribuído criado em 2009 para uso no LinkedIn
  - Autores: Jay Kreps, Neha Narkhede, e Jun Rao
- Trabalha com fluxo de dados contínuos (*data stream*)
  - Dados enviados continuamente e processados à medida em que chegam ao destino.
- Permite processar e transformar grandes fluxos de dados em tempo real.
- Baseado em eventos.

# Quem usa ?

---



# Confluent

---

- Oferece serviços de nuvem
- Baseado no Kafka
- Fundado por um dos criadores do Kafka (Jay Kreps)



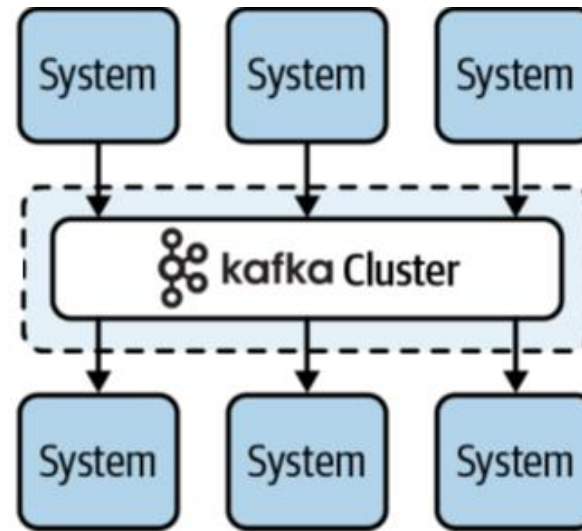
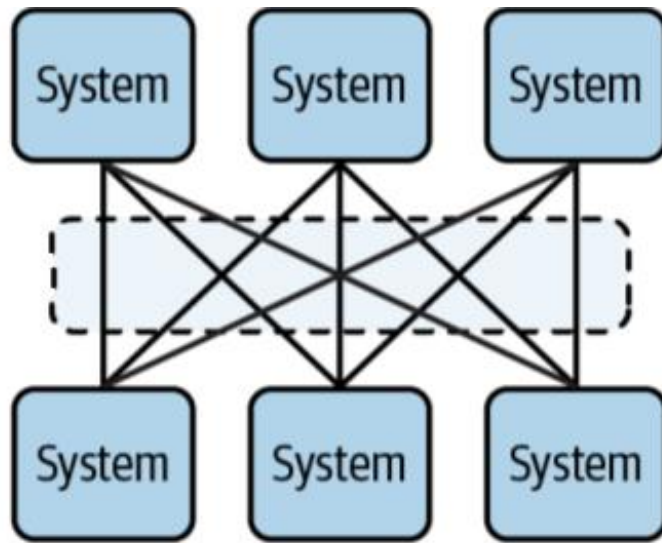
# Exemplos de geradores de dados contínuos

---

- Sensores IOT
- Monitoramento – Ex: Sensores médicos
- Sistemas financeiros – mudanças de valores de ações
- Etc.



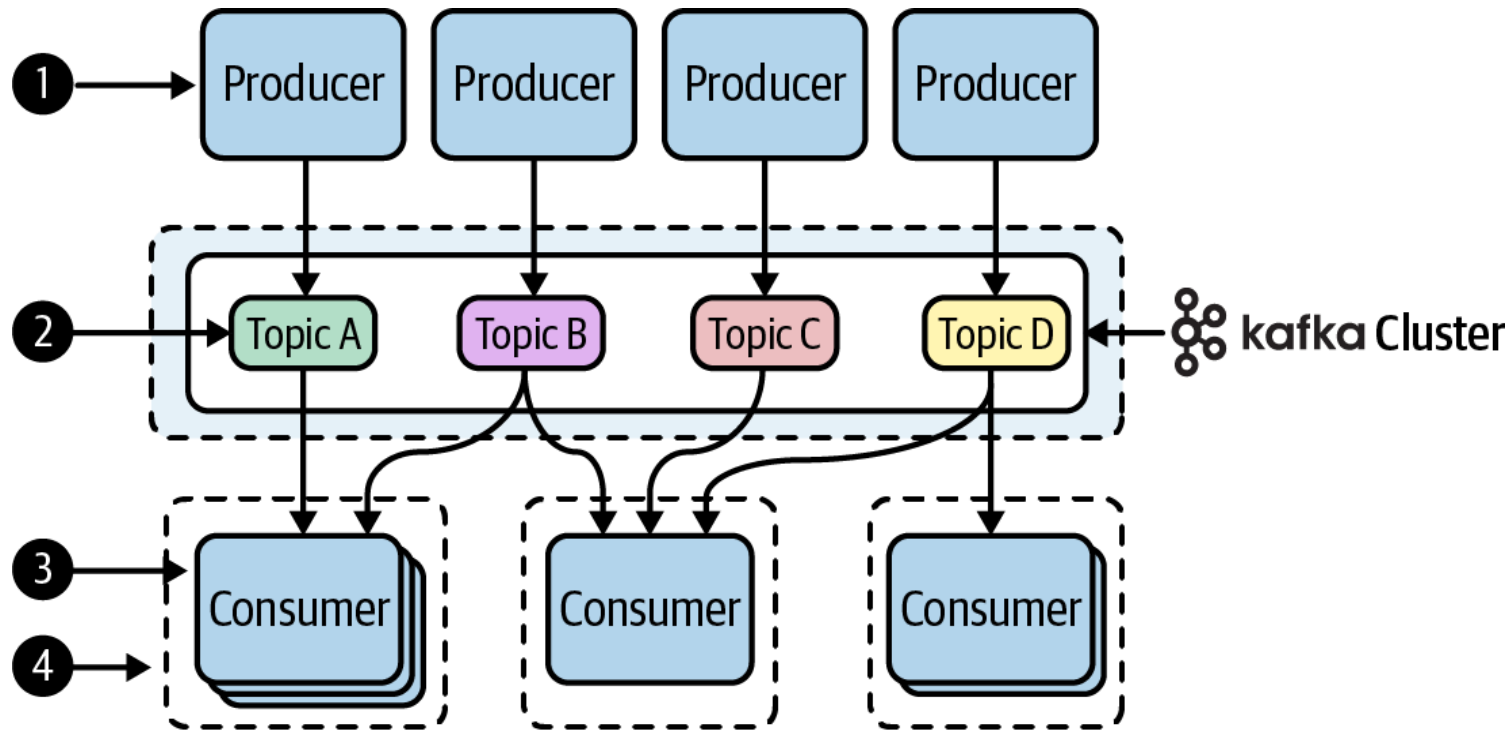
# Arquitetura Cliente Servidor x Pub/Sub



- Funciona como um *hub* de comunicação entre diferentes sistemas.
- Implementa o padrão *Publisher/Subscriber* ou simplesmente **pub/sub**

Fonte: Seymour (2021)

# Arquitetura



Fonte: Seymour (2021)

- 1) **Produtores** enviam dados em um **tópico**, sem se importar em quem vai ler
- 2) **Tópicos** agrupam informações de fluxos (*streams*) dados relacionados (similar a tabelas mas sem schema)
- 3) **Consumidores** (ou assinantes) lêem dados de um ou mais tópicos
- 4) Os consumidores podem ser agrupados (*consumer group*) para distribuir a carga de trabalho em múltiplos processos

# Arquitetura - Características

---

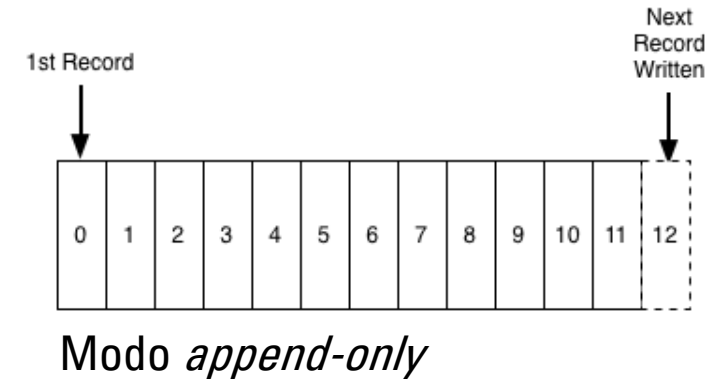
**Desacoplamento:** Os produtores e consumidores não precisam conhecer seus processos internos

Comunicação **assíncrona** e **unidirecional**

Se os consumidores não conseguem processar os dados, o Kafka funciona como um *buffer* que armazena os dados de forma durável até os consumidores conseguirem ler.

# Armazenamento de Streams - Logs

- **Logs:** Capturam uma sequência ordenada de eventos
- Não confundir com logs de aplicação
- Cada registro no Log é imutável

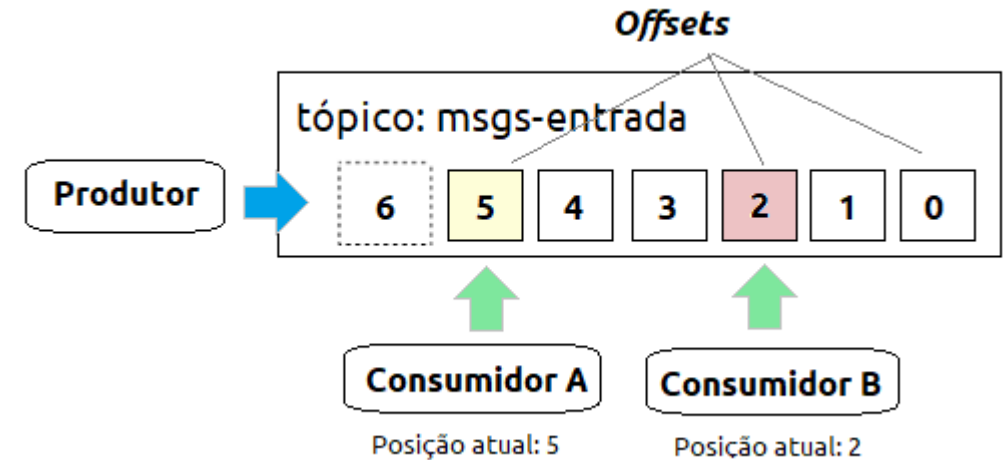


```
Windows PowerShell  ubuntu
→ ~ cat users.log
timestamp=1597373669,user_id=1,purchases=1
timestamp=1597373669,user_id=2,purchases=1
timestamp=1597373669,user_id=3,purchases=1
timestamp=1597373669,user_id=4,purchases=1
timestamp=1597374265,user_id=1,purchases=2
→ ~
```

Exemplo: Se o mesmo cliente faz mais de uma compra, Insiro um novo registro e não atualizo o original

# Armazenamento de Streams – Logs - Offset

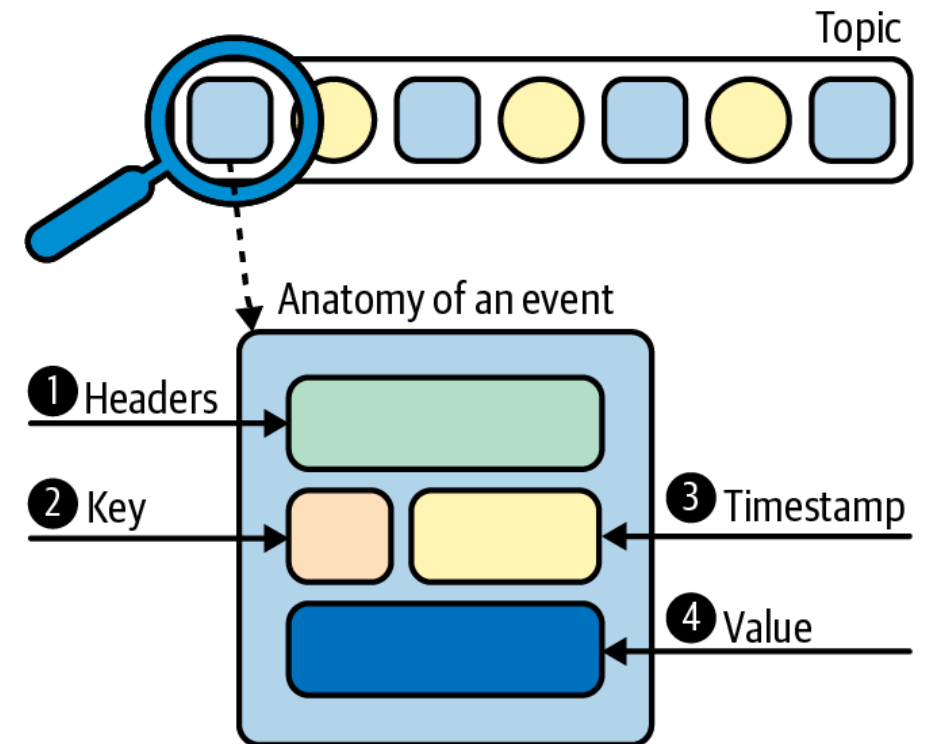
- *Offset* (deslocamento): Mantém um registro de cada entrada por meio de um índice.
- Exemplo: Enquanto o produtor está gravando um novo registro no Kafka no tópico FIAP, as aplicações consumidoras A e B estão lendo mensagens em posições diferentes de acordo com suas capacidades de processamento.



# Eventos

**Evento** no Kafka é um **registro de que algo aconteceu** e pode conter:

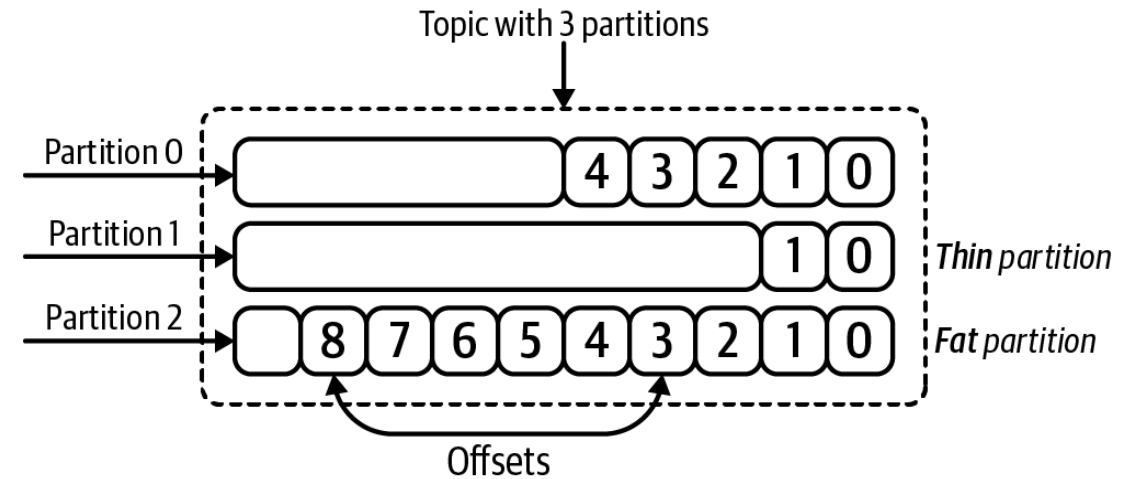
- 1) **Cabeçalhos**: Metadados da mensagem
- 2) **Chave** (Opcional): Um tópico pode opcionalmente incluir uma ou mais chaves. Os consumidores poderão ler de chaves específicas.
- 3) **Timestamp**: registro de data e hora em que ocorreu o evento
- 4) **Valor**: Conteúdo da mensagem (*payload*). Ex: Conteúdo JSON.



Fonte: Seymour (2021)

# Partições

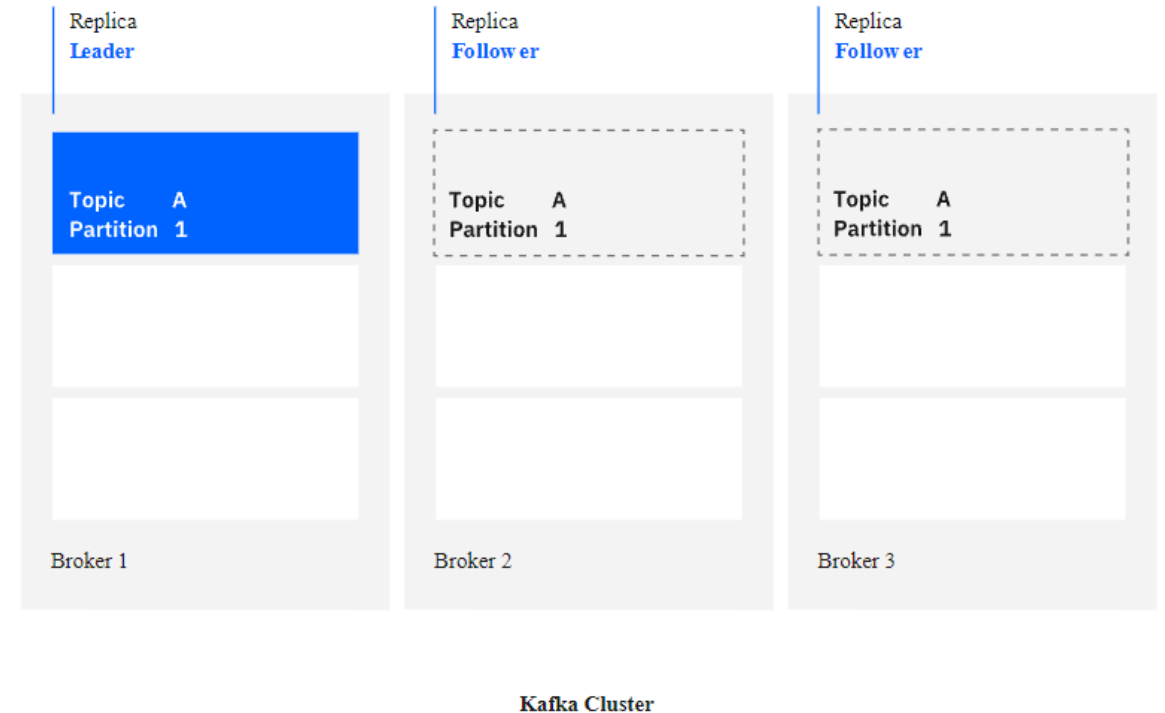
- Um **tópico** é composto de uma ou mais *partições*. As mensagens em uma partição são ordenadas (*offset*).
- Ao ter **múltiplas partições**, é possível **aumentar a escalabilidade**.
- Como a ordenação é realizada dentro de cada partição, alguns produtores podem optar por gravar algumas mensagens (determinadas chaves) em uma mesma partição.



Fonte: Seymour (2021)

# Brokers e Clusters

- *Brokers* são nós (máquinas) que **armazenam e distribuem as mensagens** de cada tópico e suas partições.
- Um grupo de  $n$  máquinas que armazenam as mensagens de um tópico de forma distribuída é chamado de *cluster*.
- Permitem replicação e alta disponibilidade.
- Obs: é possível ter apenas 1 broker. Recomenda-se ao menos 3 *brokers* em ambientes de produção.

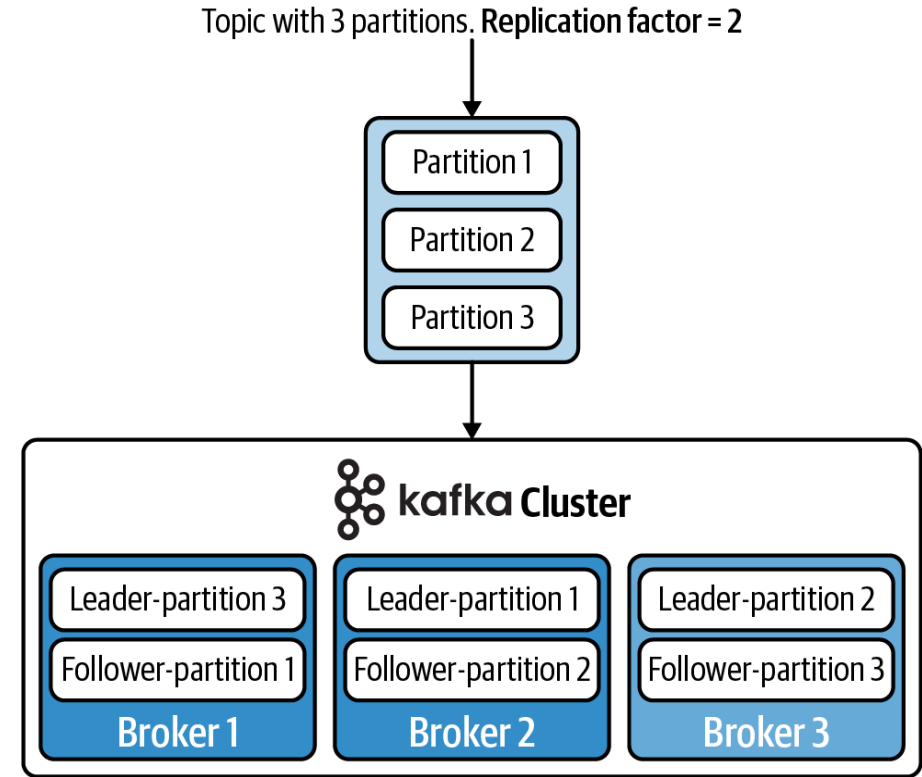


Fonte: IBM



# Replicação

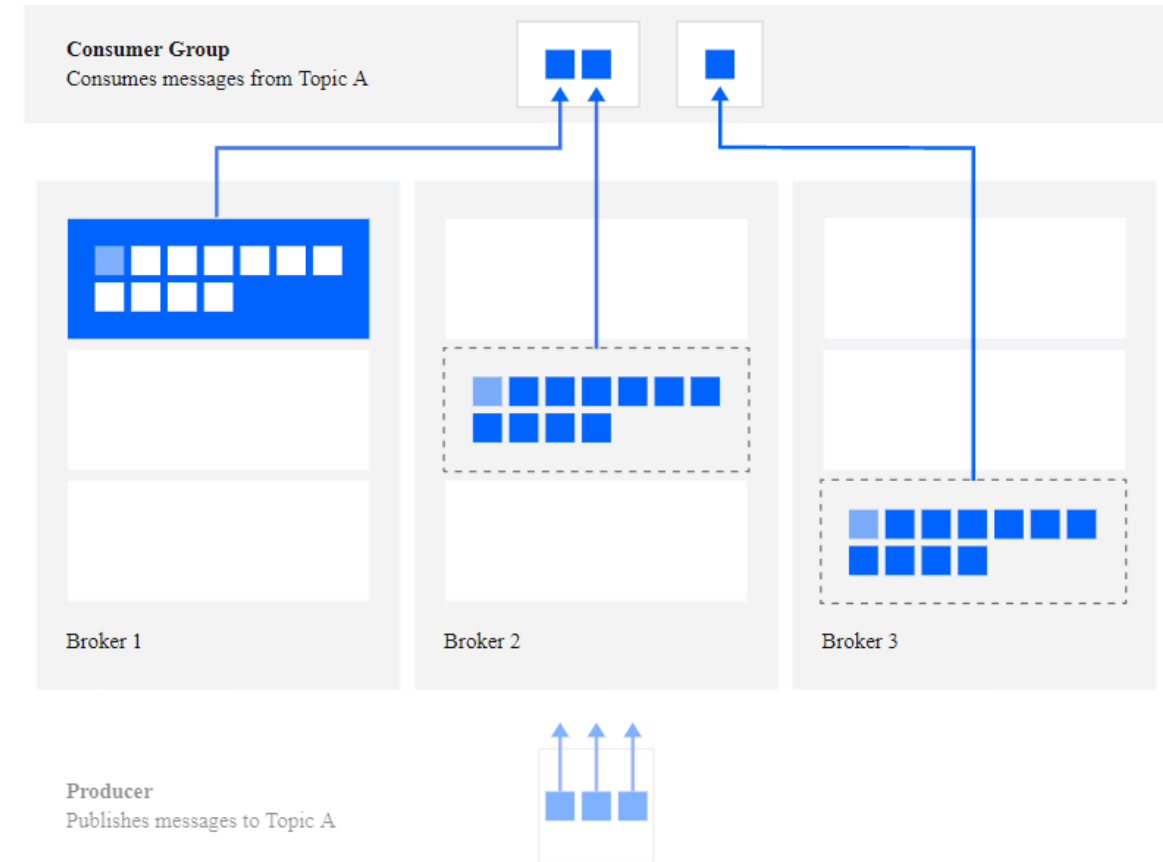
- Os *produtores* gravam na partição líder. Os *seguidores* buscam repetidamente no líder.
- Em caso de falha de um produtor, um seguidor é promovido a líder.
- É possível adicionar novos brokers em caso de aumento de carga
- O fator de replicação determina em quantas máquinas uma partição é replicada.



Fonte: Seymour (2021)

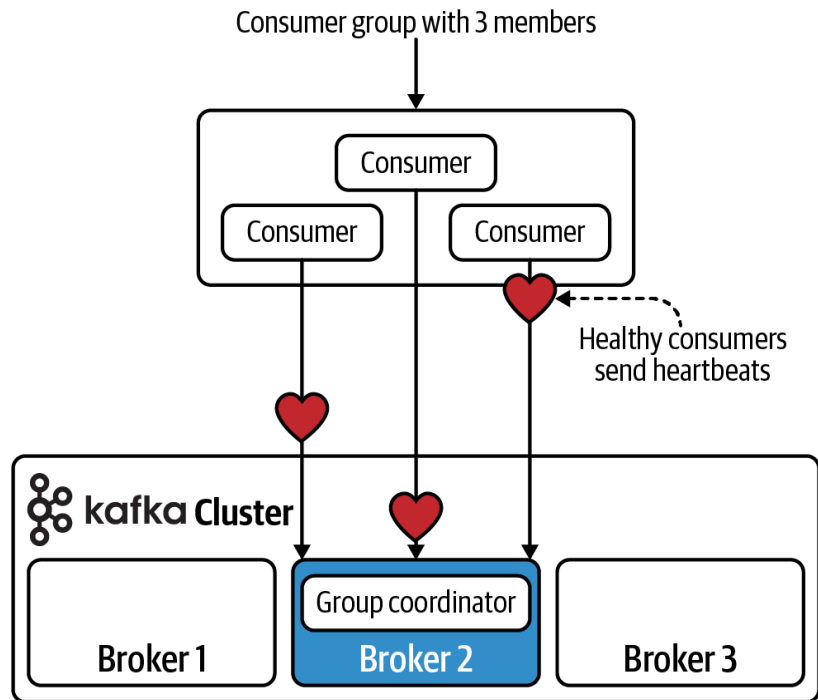
# Grupos de Consumidores

- Para agilizar a leitura e processamento das mensagens, é possível criar grupos de consumidores.
- É possível adicionar novos consumidores ao grupo para aliviar o processamento.
- Se um consumidor falhar, outro assume a carga de leitura.



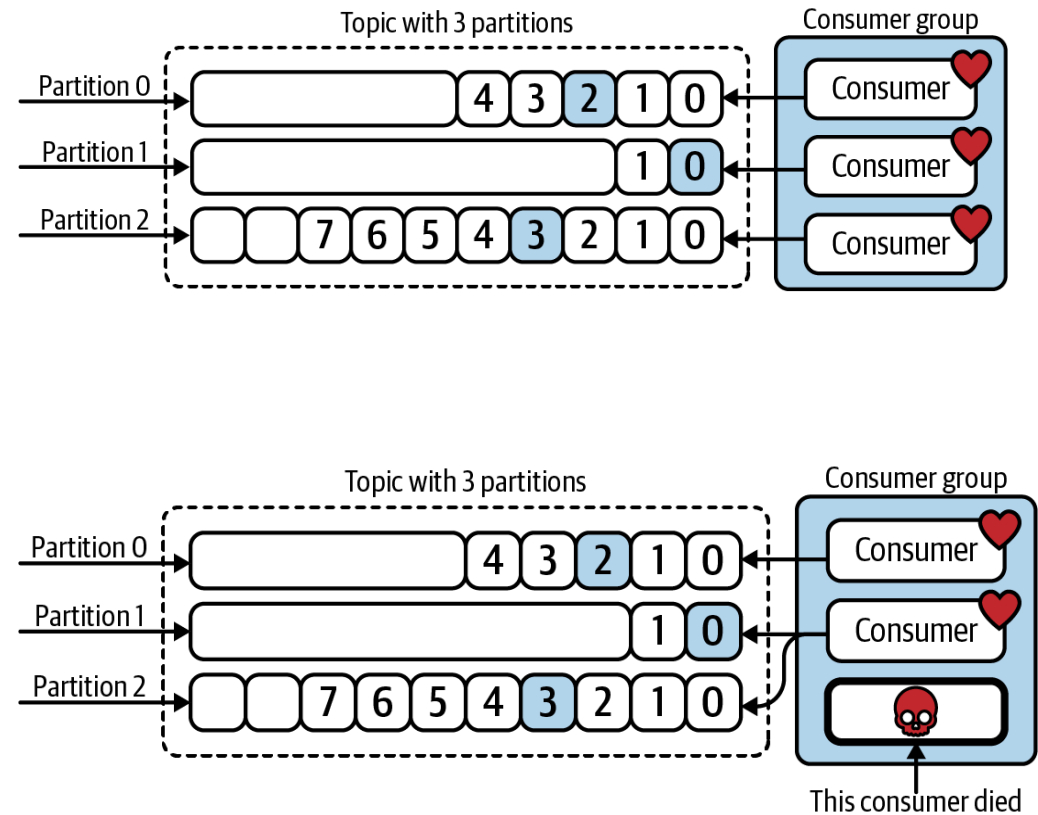
Fonte: IBM

# Grupos de Consumidores



Fonte: Seymour (2021)

- O coordenador de grupos recebe informações de que os consumidores estão saudáveis (*heartbeat*)
- Rebalanceia a carga em caso de falha de um dos consumidores



# AWS MSK – Managed Streaming for Kafka

- Cria um cluster (VMs no EC2) para uso em aplicações que já utilizam o kafka
- As aplicações podem criar tópicos e ler diretamente do MSK.
- Ideal para aplicações que já utilizam o Kafka e que precisam migrar para a nuvem.
- Permite armazenar os dados em tempo ilimitado.

The screenshot displays the AWS MSK console interface. On the left, a navigation sidebar shows 'Amazon MSK' with a close button, and a list of options: 'Clusters MSK' (expanded), 'Clusters', 'Configurações de cluster', 'MSK Connect', 'Conectores', and 'Plug-ins personalizados'. The main content area is titled 'Amazon MSK > Clusters'. It features a 'Clusters' section with a search bar labeled 'Encontrar clusters', a filter dropdown set to 'Todos os...', and a 'Criar cluster' button. Below this is a table with columns: 'Nome do cluster', 'Status', 'Tipo de cluster', 'Autenticação', and 'Versão ...'. The table is currently empty. A red 'ITV' logo is visible in the bottom right corner of the console area.

# AWS Kinesis – Analisa streams em tempo real

---

- Serverless
- Opera de forma similar com o Kafka
- Permite analisar streams em tempo real (Realtime analytics) na nuvem.
- Suporte a video streams.
- Data Firehose – Processa e entrega dados em tempo real
  - Ex: entrega os dados transformados para o S3, MongoDB, Redshift, etc.
- Data Analytics – Aplicativo de análise de dados
- Mais restrito à AWS. Difícil migração para outros vendedores.

# AWS Kinesis – Analisa streams em tempo real

---

Analytics

## Serviços do Amazon Kinesis

Colete, processe e analise facilmente streams de dados em tempo real.

### Comece a usar

- ☒ Kinesis Data Streams  
Colete dados de streaming com um fluxo de dados.
- ☐ Kinesis Data Firehose  
Processe e entregue dados de streaming com o fluxo de entrega de dados.
- ☐ Kinesis Data Analytics  
Analise dados de streaming com o aplicativo de análise de dados

[Criar fluxo de dados](#)

# AWS Kinesis – Criar fluxo de dados

## Criar fluxo de dados [Info](#)

### Configuração do fluxo de dados

Nome do fluxo de dados

ftf-meu-stream

Os caracteres aceitáveis são letras maiúsculas e minúsculas, números, sublinhados, hifens e pontos.

Configuração	Valor	Editável após a criação
Modo de capacidade	Sob demanda	✓ Sim
Período de retenção de dados	1 dia	✓ Sim
Criptografia no servidor	Desabilitado	✓ Sim
Monitorar métricas aprimoradas	Desabilitado	✓ Sim
Tags	-	✓ Sim

Cancelar

Criar fluxo de dados

FIAP

# AWS Kinesis – Visualise os shards no Cloudshell

---

```
[cloudshell-user@ip-10-1-60-63 ~]$ aws kinesis describe-stream --stream-name ftf-meu-stream
{
  "StreamDescription": {
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "StartingHashKey": "0",
          "EndingHashKey": "85070591730234615865843651857942052863"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber": "49633292692582484160415954031522168644611629533349543938"
        }
      },
      {
        "ShardId": "shardId-000000000001",
        "HashKeyRange": {
          "StartingHashKey": "85070591730234615865843651857942052864",
          "EndingHashKey": "170141183460469231731687303715884105727"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber": "49633292692604784905614484654663704362884277894855524370"
        }
      },
      {
        "ShardId": "shardId-000000000002",
        "HashKeyRange": {
```





# AWS Kinesis – Insira dados em uma partição

---

```
}  
[cloudshell-user@ip-10-1-60-63 ~]$ aws kinesis put-record \  
> --stream-name ftf-meu-stream \  
> --data '{"nome":"Fernando"}' \  
> --partition-key 2 --cli-binary-format raw-in-base64-out  
{  
  "ShardId": "shardId-000000000003",  
  "SequenceNumber": "49633292692649386396011545902291101310841152073864577074"  
}
```

# AWS Kinesis – Ler TODOS os dados do stream

aws kinesis get-shard-iterator --stream-name ftf-meu-stream --shard-iterator-type TRIM\_HORIZON --shard-id shardId-000000000003

```
[cloudshell-user@ip-10-1-60-63 ~]$ aws kinesis get-shard-iterator --stream-name ftf-meu-stream --shard-iterator-type TRIM_HORIZON --shard-id shardId-000000000003
{
  "ShardIterator": "AAAAAAAAAFAHFQ0vZFFZwPj2pu06ZUZt2NuUwcwKCSru/Kax3KBt62JP/k1GUCAqny3TeoxtbxhU8nf9aKN8S/MsoiRzCY1MDHaMFsZ1F9K7TJjyLp5NopN7lGi2vh3M7uRhF8yqLCEi4JhhyYw1FNcIamQgWrltrcNrIqvbZuStf1H3FSaVBafdeHcMFPoQwUB0dyz9uQohtwrsKv1Ai1dme5B7fcz0GrQQTp7pHR4yXuI4yoTNw=="
}
```

```
[cloudshell-user@ip-10-1-60-63 ~]$ aws kinesis get-records --shard-iterator AAAAAAAAAAFAHFQ0vZFFZwPj2pu06ZUZt2NuUwcwKCSru/Kax3KBt62JP/k1GUCAqny3TeoxtbxhU8nf9aKN8S/MsoiRzCY1MDHaMFsZ1F9K7TJjyLp5NopN7lGi2vh3M7uRhF8yqLCEi4JhhyYw1FNcIamQgWrltrcNrIqvbZuStf1H3FSaVBafdeHcMFPoQwUB0dyz9uQohtwrsKv1Ai1dme5B7fcz0GrQQTp7pHR4yXuI4yoTNw==
{
  "Records": [
    {
      "SequenceNumber": "49633292692649386396011545902287474533382293961408774194",
      "ApproximateArrivalTimestamp": "2022-09-14T13:19:39.731000+00:00",
      "Data": "dGVzdGU=",
      "PartitionKey": "1"
    },
    {
      "SequenceNumber": "49633292692649386396011545902288683459201911545520980018",
      "ApproximateArrivalTimestamp": "2022-09-14T13:20:22.397000+00:00",
      "Data": "dGVzdGU=",
      "PartitionKey": "1"
    },
    {
      "SequenceNumber": "49633292692649386396011545902291101310841152073864577074",
      "ApproximateArrivalTimestamp": "2022-09-14T13:23:06.165000+00:00",
      "Data": "eyJub21lIjoiRmVybmlFuZG8ifQ==",
      "PartitionKey": "2"
    }
  ]
}
```


# AWS Kinesis – Decodifique a msg de base64

<https://www.base64decode.org/>

## Decode from Base64 format


Simply enter your data then push the decode button.



```
eyJub21lljoiRmVybmFuZG8ifQ==
```

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 **DECODE**  Decodes your data into the area below.

```
{"nome":"Fernando"}
```

FIAP

# AWS Kinesis – Leia a última msg

---

aws kinesis get-shard-iterator --stream-name ftf-meu-stream --shard-iterator-type LATEST --shard-id shardId-0000000000003

```
[cloudshell-user@ip-10-1-60-63 ~]$ aws kinesis get-shard-iterator --stream-name ftf-meu-stream --shard-iterator-type LATEST --shard-id shardId-0000000000003
{
  "ShardIterator": "AAAAAAAAAAG4ZCNbkWmU49UATwazXXhNAHHIKSZtF3WEH06kvztHSMSTieyWaLRE567hnGZE7S0/JuXw5ZyHCkIc/U9TrfMyCDoIeA//JkBrrbLEwpDwL1uFcJ+k2IAZN/OyBAsLujZUU+NXWv7tIwOc9jt9NEicLPBjgN1Cry4tqo5f/SeyfEmLPEleGY7fMWcLWsHGZKGN4Zz3gq0odjkdZQ=="
}
```

# Leituras Complementares

---

- *The Log: What every software engineer should know about real-time data's unifying abstraction* - <https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying>

# Tarefa – Criar um data stream

- ☐ No AWS Learner Lab, faça:
- ☐ Acesso o AWS Kinesis
- ☐ Crie um data stream no padrão <iniciais>-meu-stream
- ☐ Insira um registro neste stream pelo aws cloud shell
- ☐ Leia o registro e decodifique-o
- ☐ **Encerre o lab**

# Obrigado!