# Apache Kafka

## desde una perspectiva Java Dev

Victor Ramos Huarachi
Confluent Certified Developer for Apache Kafka
victor.ramos.h@gmail.com
@vramosh

# Contenido

Qué problemas se resuelve con Kafka?

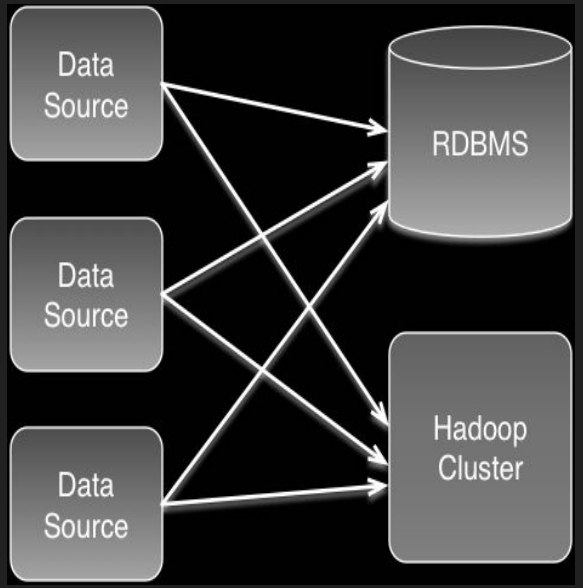
Qué es Kafka?

Principales componentes

Mi experiencia

# Qué problemas se resuelve con Kafka?



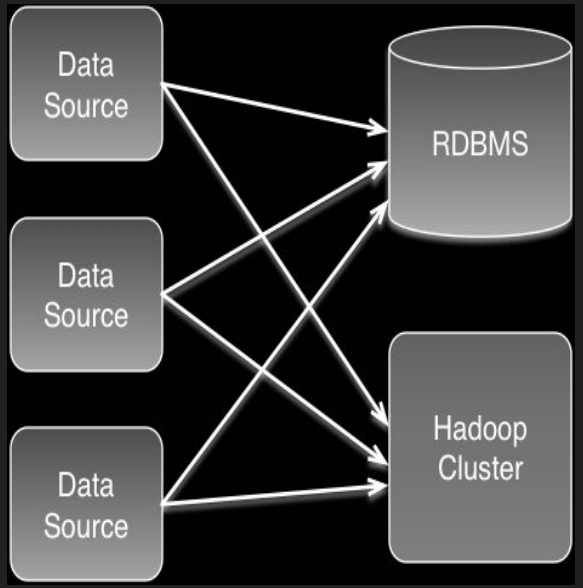


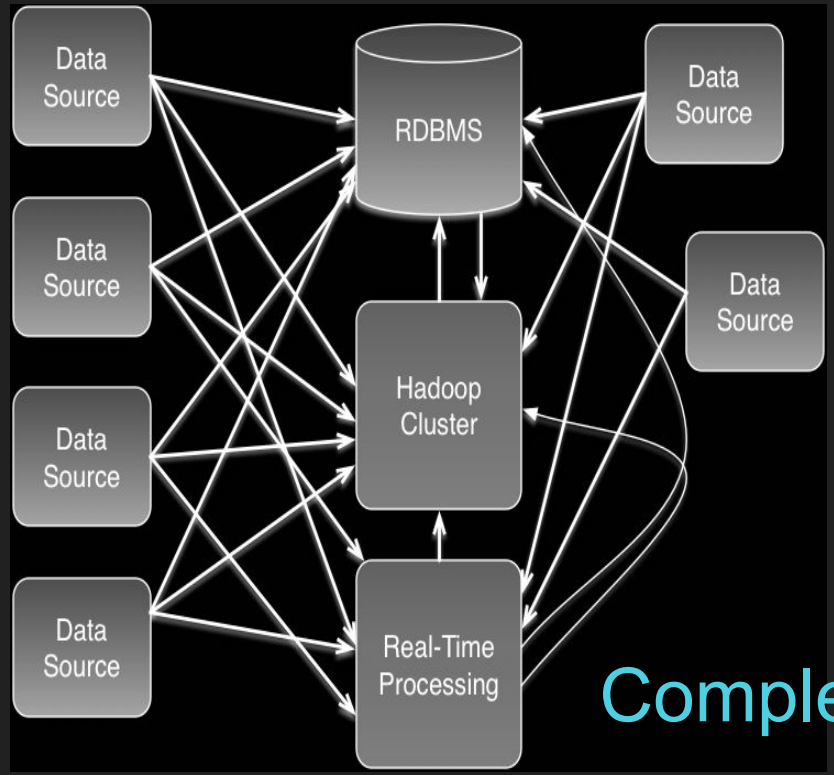

Todo sistema es simple

… al principio

# Qué problemas se resuelve con Kafka?

1




Complejo !!!

# Dificultades a nivel de desarrollo de aplicaciones

- Para 6 fuentes de datos y 3 objetivos tienes que escribir 18 conexiones!!!

# Dificultades a nivel de desarrollo de aplicaciones

- Para 6 fuentes de datos y 3 objetivos tienes que escribir 18 conexiones!!!

- Cada conexión es particular en cuanto a protocolos (TCP, FTP, HTTP, JDBC, etc) y formatos (CSV, XML, JSON, Binary, etc)

# Dificultades a nivel de desarrollo de aplicaciones

- Para 6 fuentes de datos y 3 objetivos tienes que escribir 18 conexiones!!!

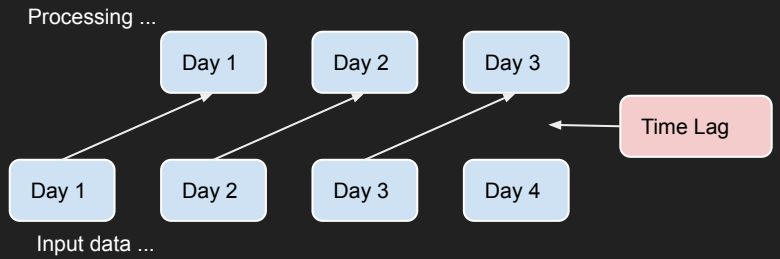
- Cada conexión es particular en cuanto a protocolos (TCP, FTP, HTTP, JDBC, etc) y formatos (CSV, XML, JSON, Binary, etc)

- Mientras más objetivos tengas, más carga tendrán los sistemas.

# Qué problemas se resuelve con Kafka?



**Batch processing**

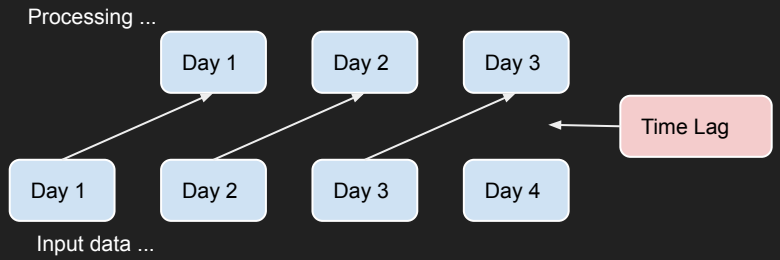



- Tradicionalmente, casi toda la data es procesada por lotes.
- Esto tiene limitaciones inherentes

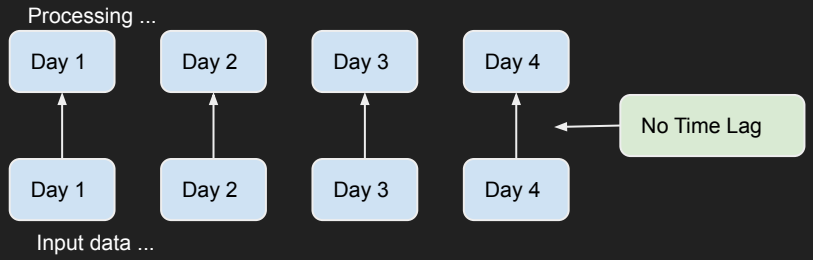
# Qué problemas se resuelve con Kafka?



## Batch processing




## Real time processing




- Tradicionalmente, casi toda la data es procesada por lotes.
- Esto tiene limitaciones inherentes
- Actualmente, a menudo es beneficioso procesar la data al momento de que esta es generada

# Simplificando la Arquitectura

# Simplificando la Arquitectura

# Qué es Kafka?

Kafka es una plataforma distribuida para streaming de datos.

Desarrollada por LinkedIn en 2011 para procesar 1.4 billones de mensajes por día.

# Qué es Kafka?

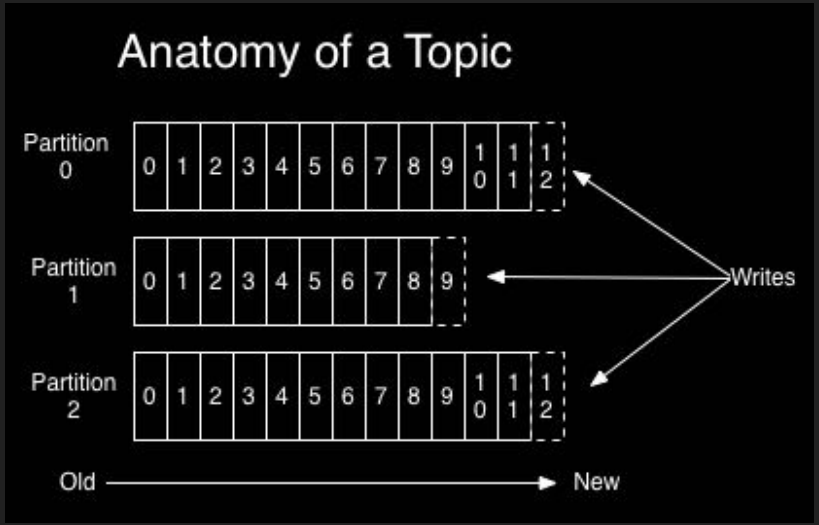
Kafka es una plataforma distribuida para streaming de datos.

Desarrollada por LinkedIn en 2011 para procesar 1.4 billones de mensajes por día.

Diseñado para resolver dos problemas:

- La simplificación de data pipelines
- Manejar streaming de datos

# Principales Componentes




## Tópicos

Son categorías donde los datos son publicados.

## Particiones

Es una secuencia  ordenada de registros que corresponden a un tópico.

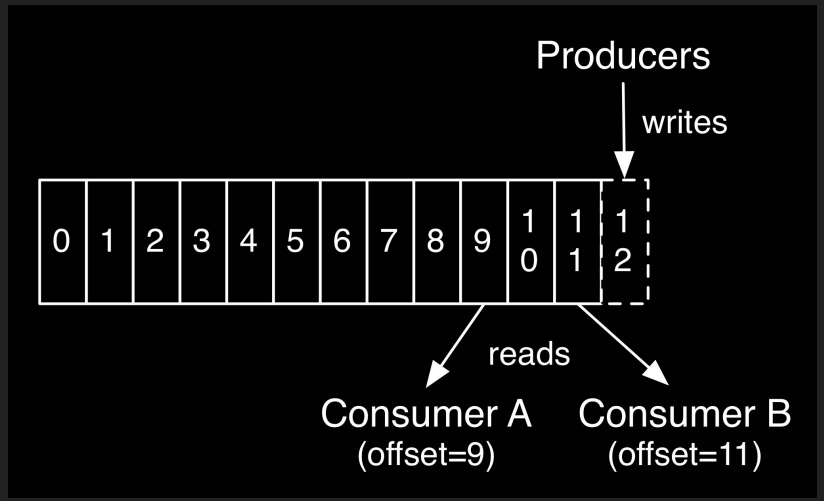
# Principales Componentes

## Productores

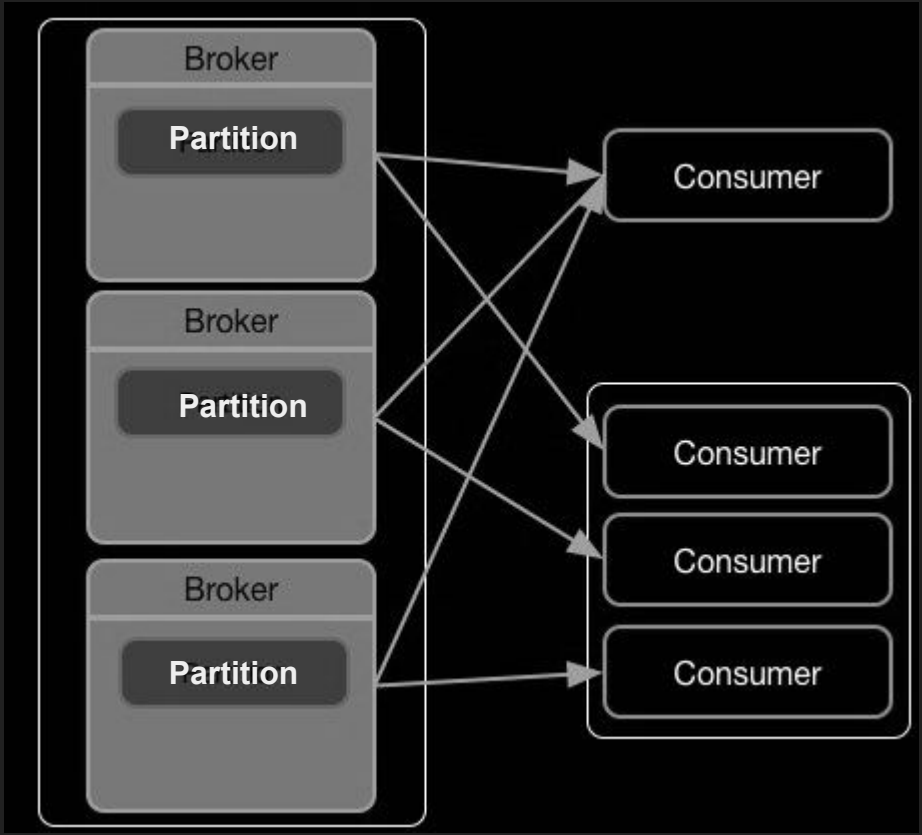
Clientes que escriben los mensajes en una determinada partición de un tópico de Kafka.

## Consumidores

Recogen los mensajes de un tópico en Kafka.



# Consumidores




IMPORTANTE !!!

1. Diferentes consumidores pueden leer datos de un mismo tópico.
2. Múltiples consumidores pueden ser combinados en un solo consumer group.

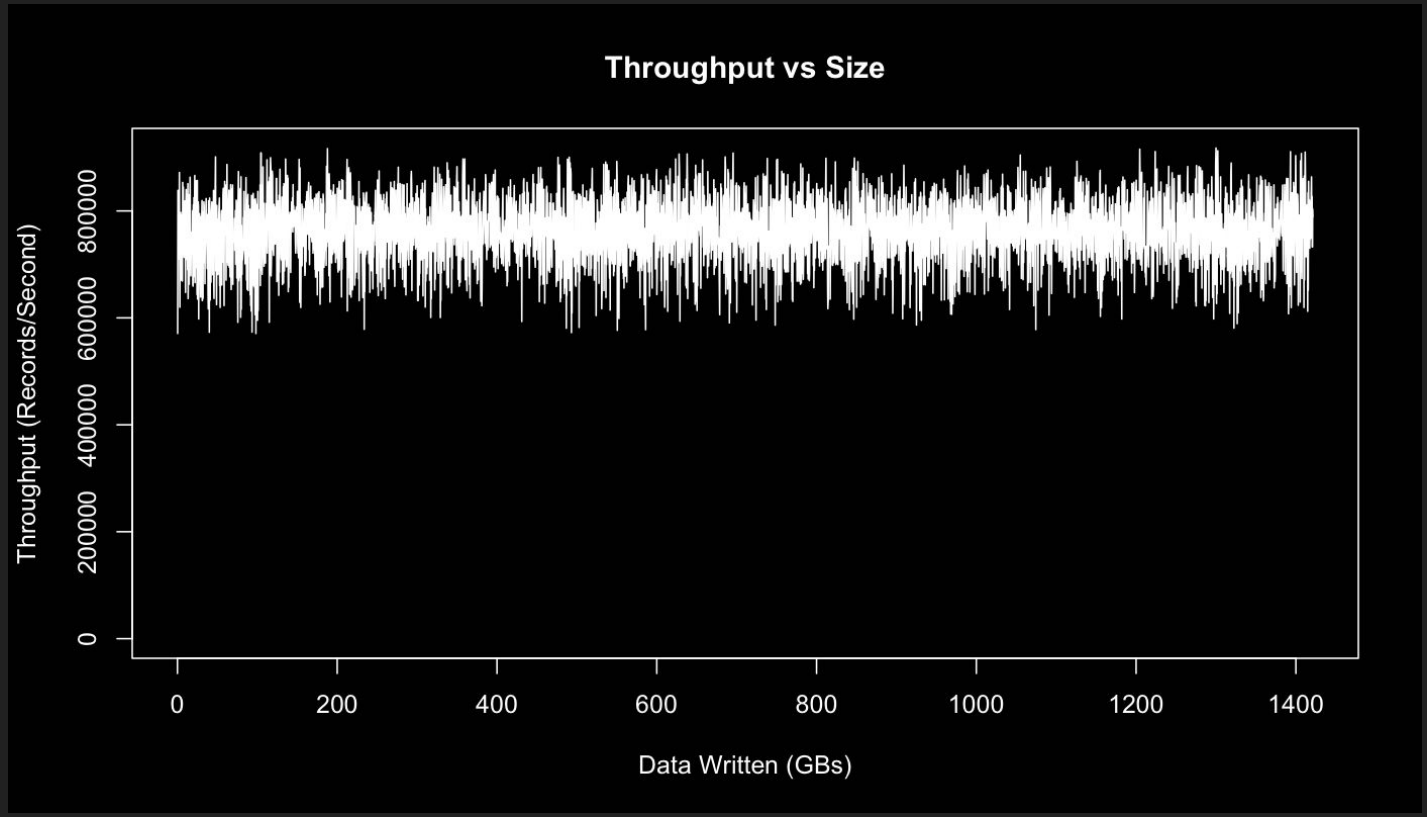
# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

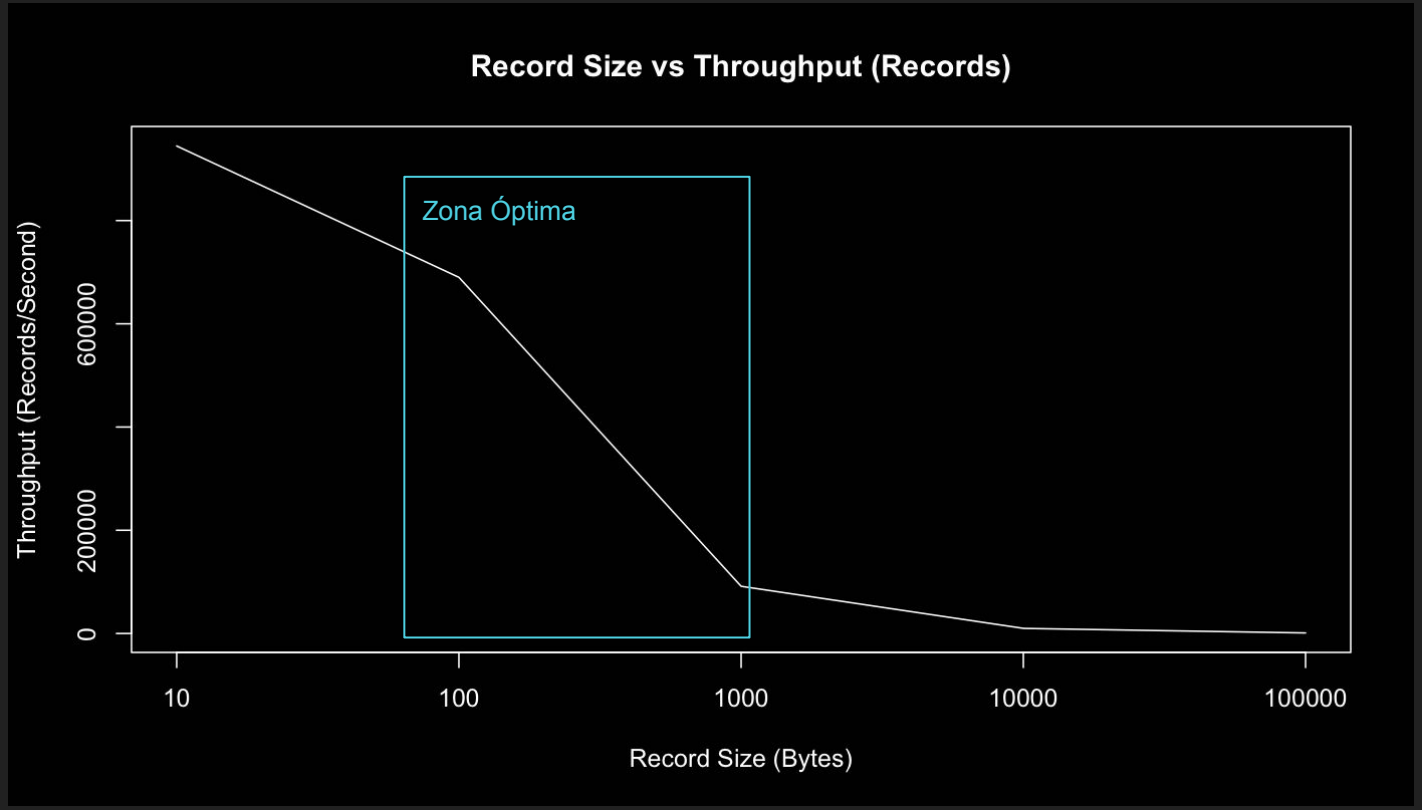
# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

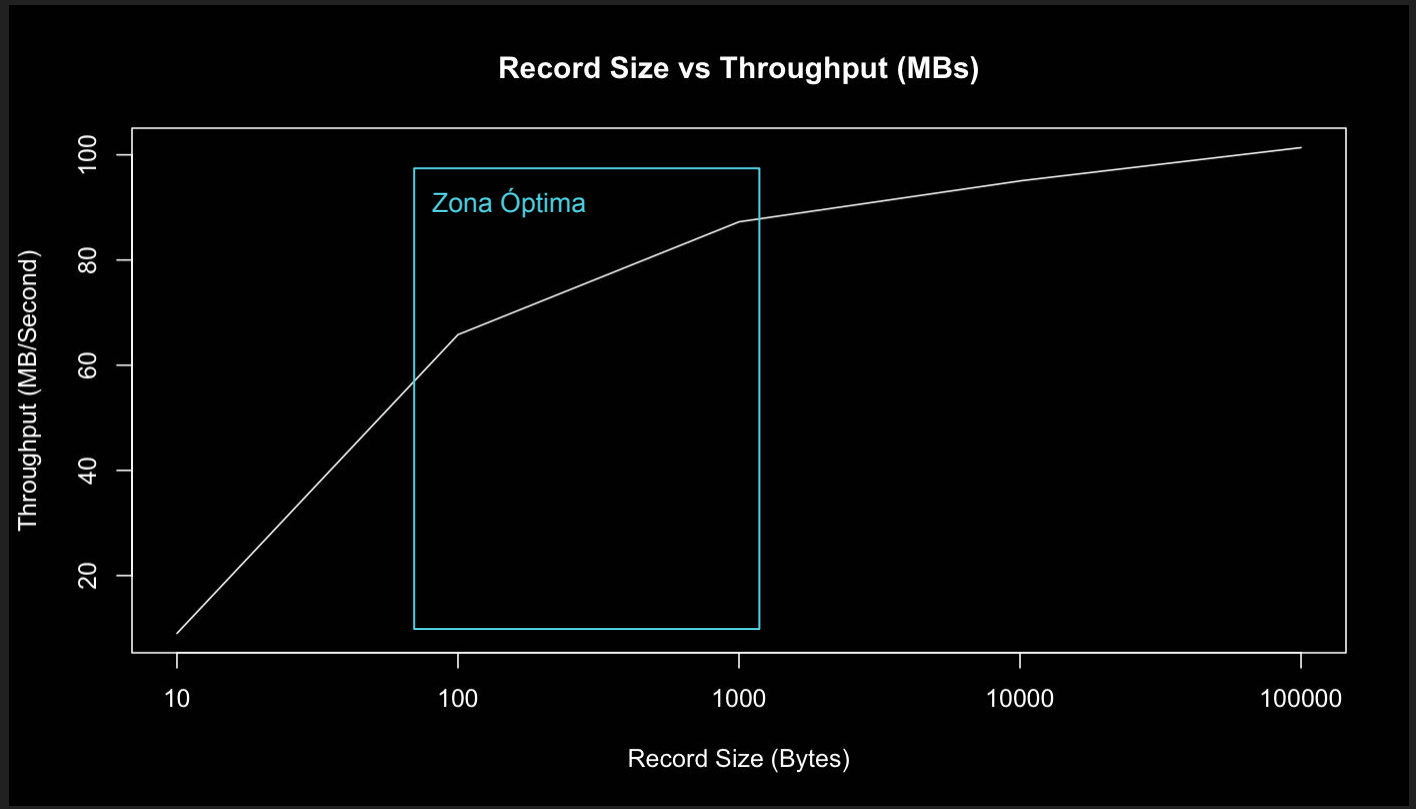
# Rendimiento de Escritura



# Cómo afecta el tamaño de un mensaje



# Cómo afecta el tamaño de un mensaje



# Mi Experiencia

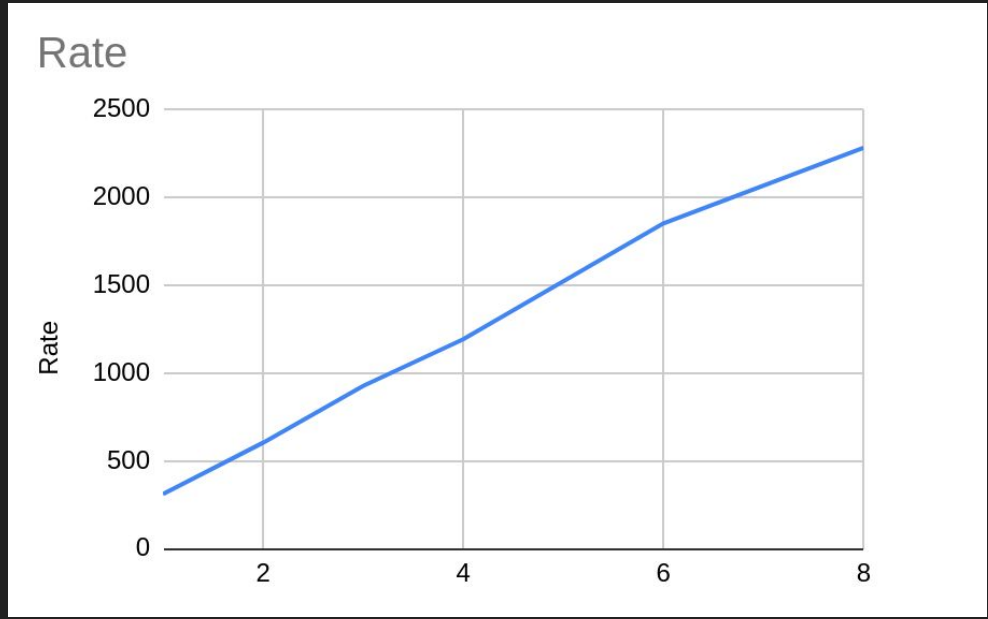

1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

# Escalabilidad

| Nro. de Instancias | Nro. de Hilos por Instancia | Rate |
|---|---|---|
| 1 | 1 | 310 |
| 1 | 2 | 602 |
| 1 | 3 | 925 |
| 1 | 4 | 1190 |
| 2 | 1 | 595 |
| 2 | 2 | 1175 |
| 2 | 3 | 1850 |
| 2 | 4 | 2280 |
| 3 | 1 | 910 |
| 3 | 2 | 1905 |



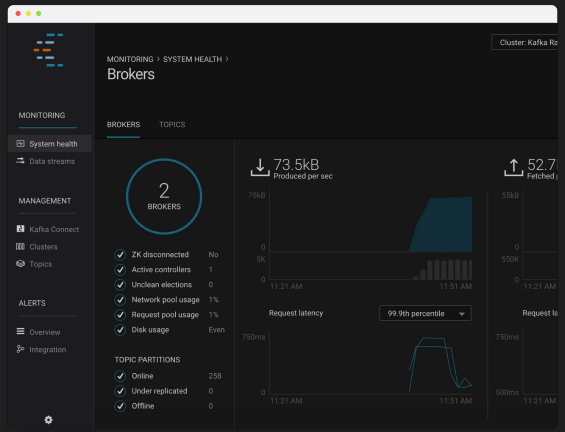
# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

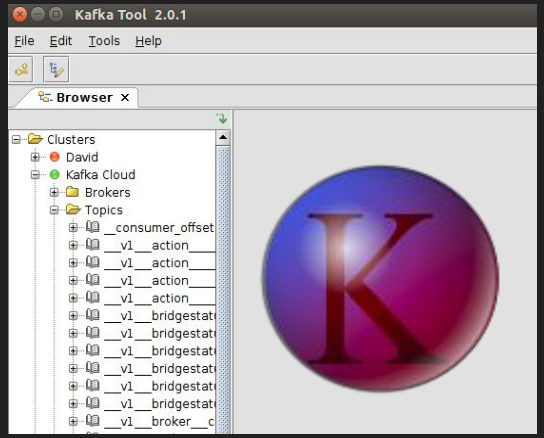
# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

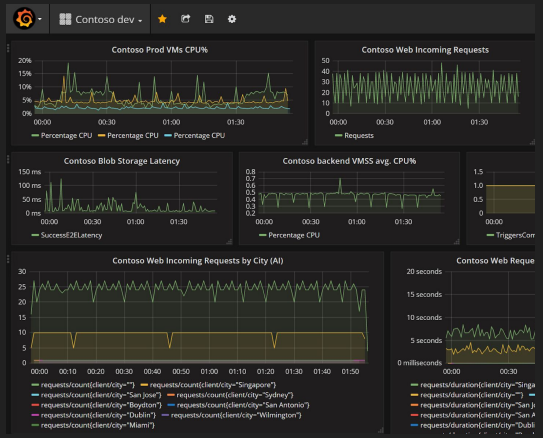
# Monitoreo




Confluent Control Centre




Kafka Tool




Kafka Exporter + Prometheus + Graphana

Command line

./kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic thetopicname --time -1 |awk -F  ":" '{sum += $3} END {print sum}'

./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --list
./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group consumergroupname

/kafka-topics.sh --zookeeper localhost:2181 --list
/kafka-topics.sh --zookeeper localhost:2181 --topic thetopicname --describe

# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

# Decide qué quieres optimizar


Durability
- Producer
  - replication.factor: 3, configure per topic
  - acks=all (default 1)
  - retries: 1 or more (default 0)
  - max.in.flight.requests.per.connection=1 (default 5)
- Broker
  - default.replication.factor=3 (default 1)
  - auto.create.topics.enable=false (default true)
  - min.insync.replicas=2 (default 1); topic override available
  - unclean.leader.election.enable=false (default true); topic override available
  - broker.rack: rack of the broker (default null)
  - log.flush.interval.messages, log.flush.interval.ms

Throughput
- Producer
  - batch.size: increase to 100000 - 200000 (default 16384)
  - linger.ms: increase to 10 - 100 (default 0)
  - compression.type=lz4 (default none)
  - acks=1 (default 1)
  - retries=0 (default 0)
  - buffer.memory: increase if there are a lot of partitions (default 33554432)
- Consumer
  - fetch.min.bytes: increase to ~100000 (default 1)

Latency
- Producer
  - linger.ms=0 (default 0)
  - compression.type=none (default none)
  - acks=1 (default 1)
- Broker
  - num.replica.fetchers: increase if followers can't keep up with the leader (default 1)
- Consumer:
  - fetch.min.bytes=1 (default 1)

Disponibilidad
- Broker
  - unclean.leader.election.enable=true (default true); topic override available
  - min.insync.replicas=1 (default 1); topic override available
  - num.recovery.threads.per.data.dir: number of directories in log.dirs (default 1)
- Consumer
  - session.timeout.ms: as low as feasible (default 10000)

# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops

# Ejemplo Rápido

https://github.com/victor-ramos-h/kafka-basics

# Mi Experiencia


1. Curva de aprendizaje
2. Rendimiento
3. Tolerancia a fallos
4. Escalabilidad
5. Concurrencia
6. Monitoreo
7. Tweaking
8. Código
9. Dev Ops