09. – 12.12.2019
Frankfurt am Main

#ittage

Bernhard Hopfenmüller

Apache Kafka -

- a system optimized for writing

Ever named a software project?

Voldemort - An evil wizard

Source: Landmark Media/ImageCollect
Project: https://github.com/voldemort/voldemort

Jay Kreps:

"I don't know whether it is nerdier to be reading Harry Potter or to be wondering what kind of consistency protocol Voldemort uses when keeping all his pieces up-to-date, but regardless, the name stuck."

Voldemort - A distributed key value store

Source: Landmark Media/ImageCollect
Project: https://github.com/voldemort/voldemort

Jay Kreps:

"I don't know whether it is nerdier to be reading Harry Potter or to be wondering what kind of consistency protocol Voldemort uses when keeping all his pieces up-to-date, but regardless, the name stuck."

Azkaban - The wizard prison

Source: https://www.bustle.com
Project: https://azkaban.github.io/

Richard Park:

"[...] clearly a phase where we started for some reason naming things after Harry Potter stuff [...]"

Azkaban - A Distributed Workflow Manager

Source: https://www.bustle.com
Project: https://azkaban.github.io/

Richard Park:

"[…] clearly a phase where we started for some reason naming things after Harry Potter stuff […]"

Kafka

Jay Kreps:

I thought that since Kafka was a system optimized for writing using a writer's name would make sense. I had taken a lot of lit classes in college and liked Franz Kafka. Plus the name sounded cool for an OS project.

Kafka - A system optimized for writing

Jay Kreps:

> *I thought that since Kafka was a system optimized for writing using a writer's name would make sense. I had taken a lot of lit classes in college and liked Franz Kafka. Plus the name sounded cool for an OS project.*

Kafka - A (distributed) system ...

Jay Kreps:

> *I thought that since Kafka was a system optimized for writing using a writer's name would make sense. I had taken a lot of lit classes in college and liked Franz Kafka. Plus the name sounded cool for an OS project.*

curl -X GET localhost://whoami

```
{"me":
  {
    "name": "Bernhard Hopfenmüller",
    "occupation":
      "Senior Consultant@ATIX AG",
    "likes and works with":
      "Automation,
       Kafka,
       Ansible,
       Python,...",
    "named a project once":
      "Supreme Octo Potato",
  }
}
```

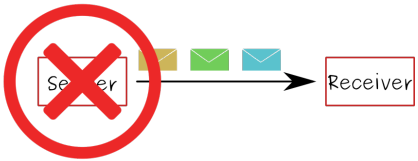
Messaging-Systems

Why do we need a messaging system? [source 1]

Messaging Systems

Why do we need a messaging system? [source 1]

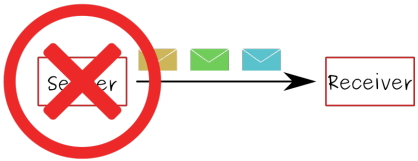
- Challenge 1: Sender not available
- Challenge 2: Sending too much (DoS)
- Challenge 3: Receiver crash upon processing



Messaging Systems

Why do we need a messaging system? [source 1]

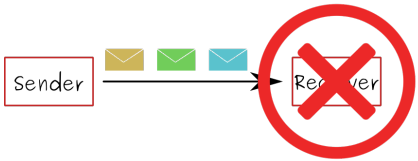
- Challenge 1: Sender not available
- Challenge 2: Sending too much (DoS)
- Challenge 3: Receiver crash upon processing



Messaging Systems

Why do we need a messaging system? [source 1]

- Challenge 1: Sender not available
- Challenge 2: Sending too much (DoS)
- Challenge 3: Receiver crash upon processing

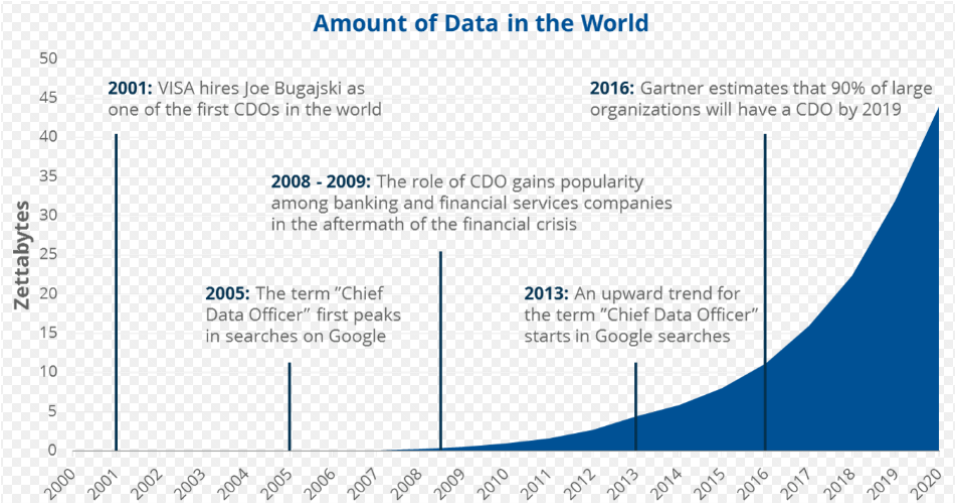


Waiting Queue vs Television [source 1]

Wait until it's your turn

Choose what you want to receive

Why do we need “Television”

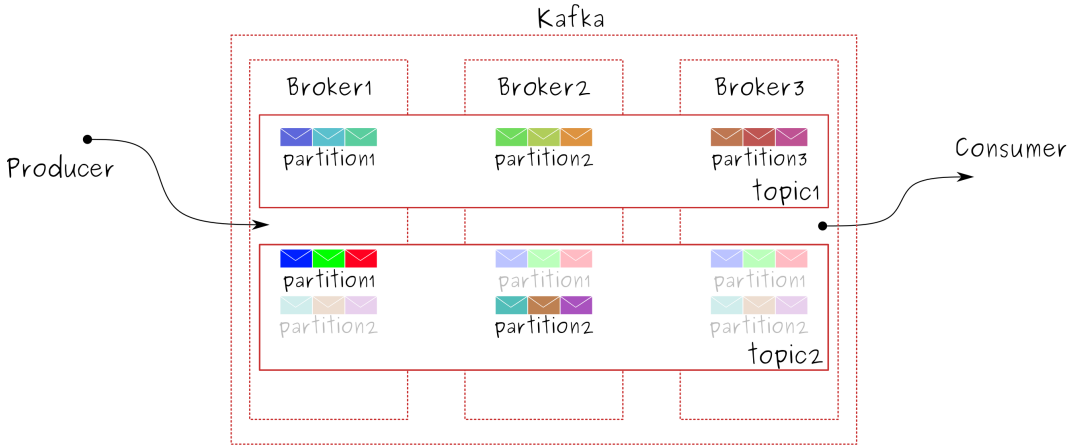


source: Javier Ramos@ medium.com

1ZB = 1000000000000000000000 bytes = 10^{21} byte

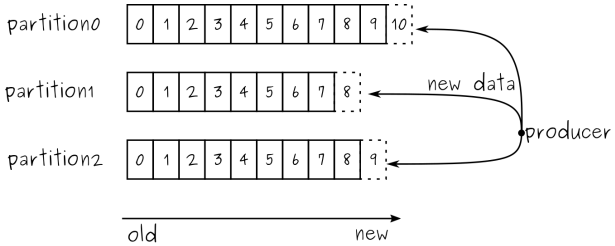
So let's watch some TV!

Kafka-Basic structure



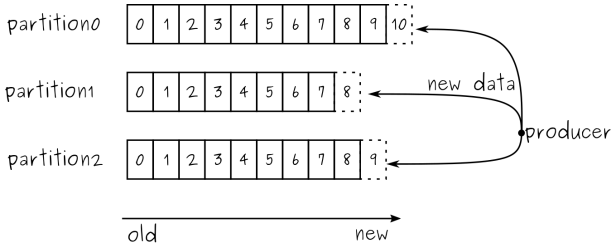
What is a topic in Kafka?

- core component of Kafka
- is filled by producer
- consists of one or more partitions

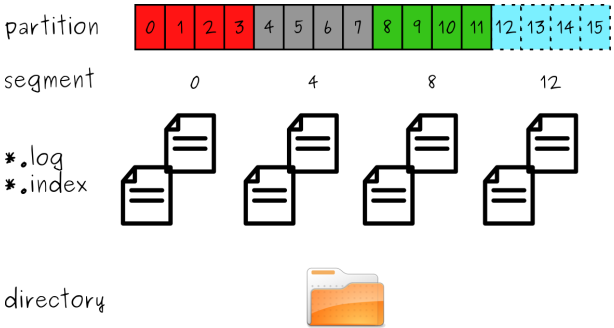


What is a topic in Kafka?

- producer can choose partition
- partition has running offset
- message is identified by offset

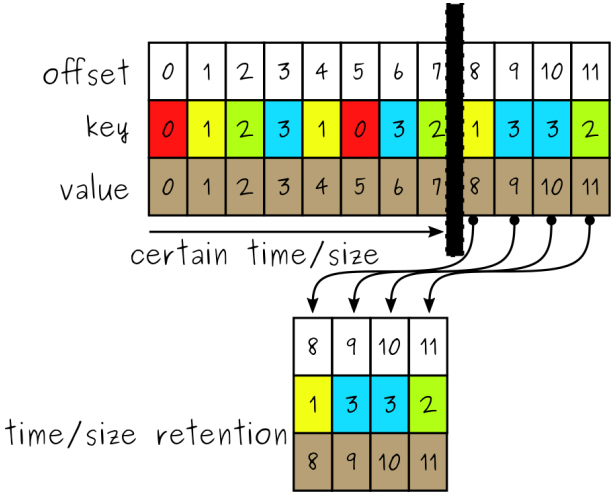


Data Storage in topics



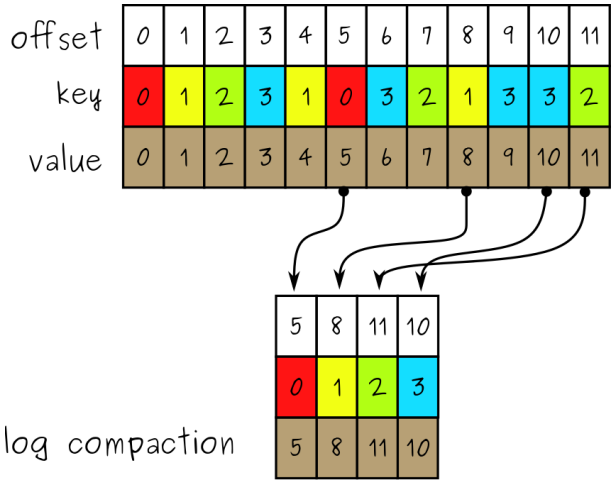
- messages are stored physically!
- key-value principle
- Clean-Up policies:

Cleaning up topics



- Clean-Up policies:
  - default: Retention-time (delete old data after *x* days)
  - Retention-size (delete old data if data memory > x)

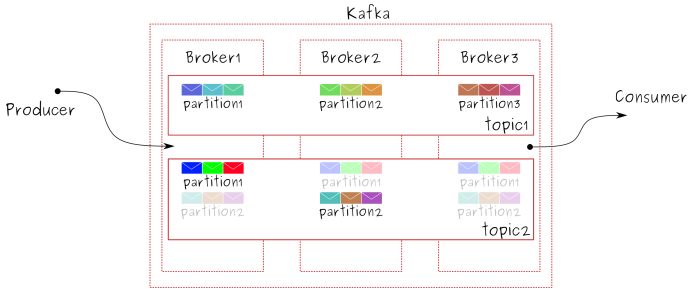
Cleaning up topics



- Clean-Up policies:
  - default: Retention-time (delete old data after *x* days)
  - Retention-size (delete old data if data memory > x)
  - Log-Compaction (replace old value to key with new)

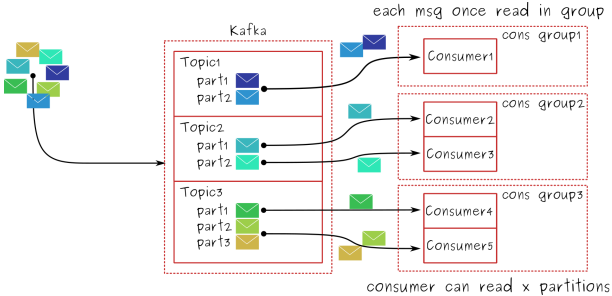
Topic consumption

- topics are pulled! (no DoS)
- any existing data can be pulled

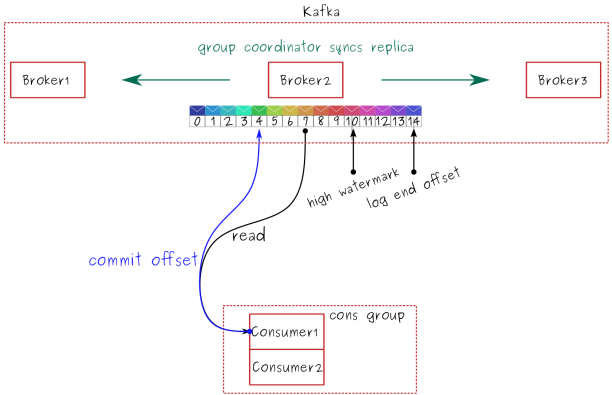


Consumer Groups

- parallelism allows high throughput
- never more consumers than partitions



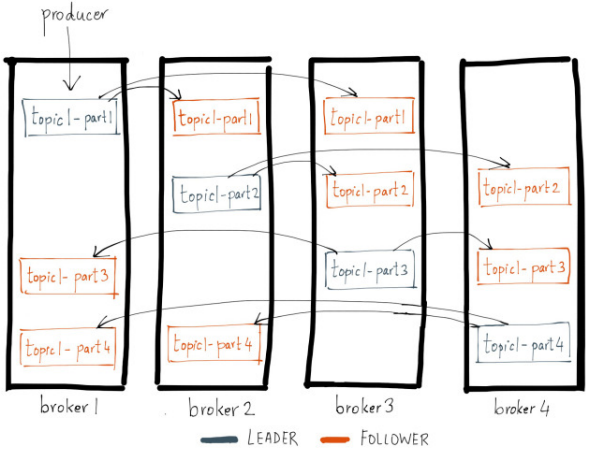
Wait but who knows what's read?



- Consumer commit their offset
- Upon failure re-processing possible

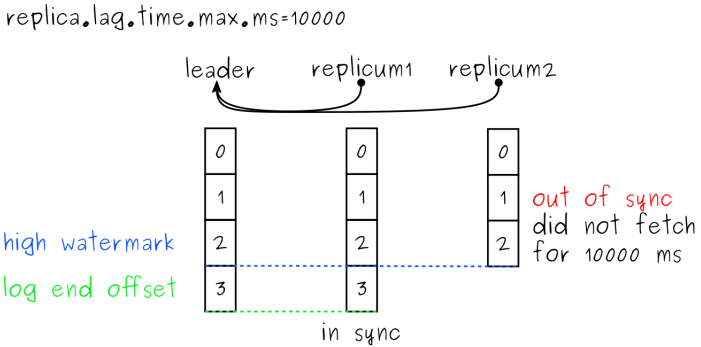
Replication

implemented on partition level



source: Confluent

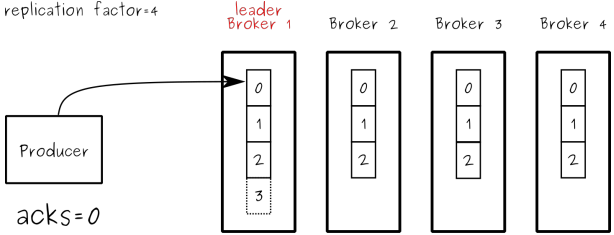
In and Out of Sync Replica



Did somebody hear my message?

Producer decides if message was successfully sent

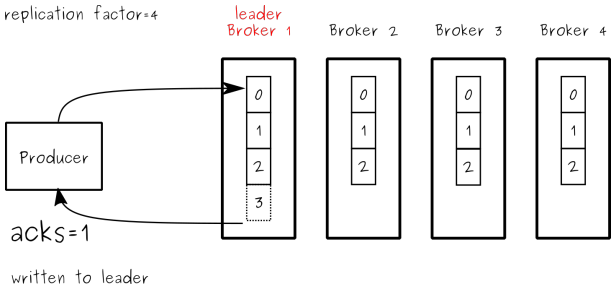
Delivery Guarantees



fire and forget, no feedback

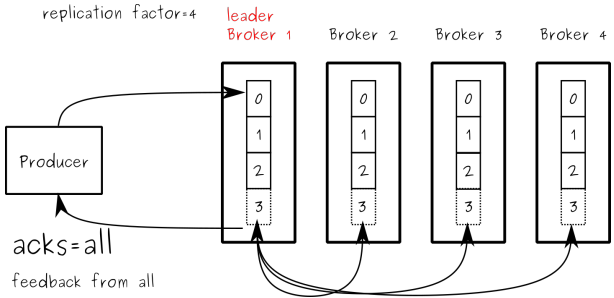
- as soon as sent
- as soon as received by first broker
- as soon as desired number of replica exist

Delivery Guarantees



- as soon as sent
- as soon as received by first broker
- as soon as desired number of replica exist

Delivery Guarantees



- as soon as sent
- as soon as received by first broker
- as soon as desired number of replica exist

Delivery Semantics

- At-most-once: Message is only delivered once or not at all
- At-least-once: Message is delivered once or more often
- Exactly-once: Message is delivered once - Kafka!

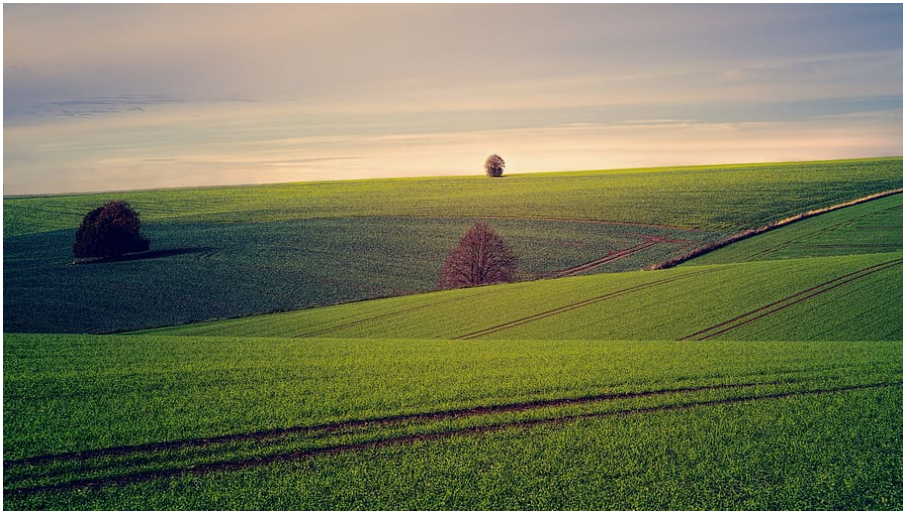
Delivery Semantics

- At-most-once: Message is only delivered once or not at all
- At-least-once: Message is delivered once or more often
- Exactly-once: Message is delivered once - Kafka!

Delivery Semantics

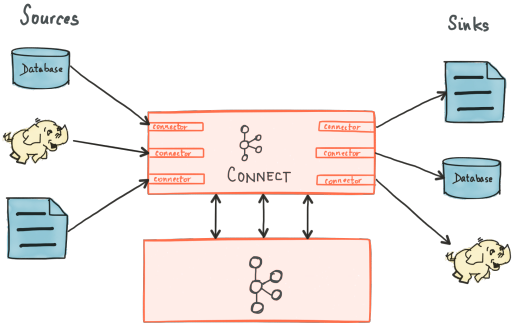
- At-most-once: Message is only delivered once or not at all
- At-least-once: Message is delivered once or more often
- Exactly-once: Message is delivered once - Kafka!

Green meadows?



Almost green meadows?

Talk to Kafka - Kafka Connect

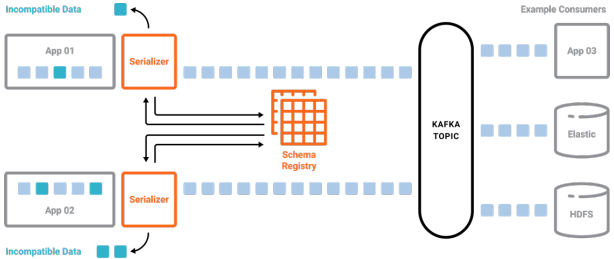


source: Confluent

- I/O for Kafka
- Connect with external/existing systems
- Open Source by Confluent

Talk to Kafka - Schema Registry

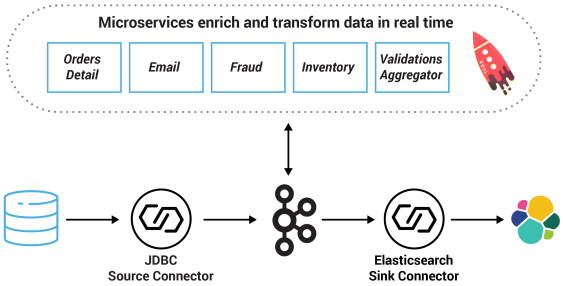
- define standards
- version and store them
- Open Source by Confluent



source: Confluent

Going further: Kafka Streams

- reading data in real-time, process it, write it back
- scale applications
- Consumers is Producer
- different API



source: Confluent

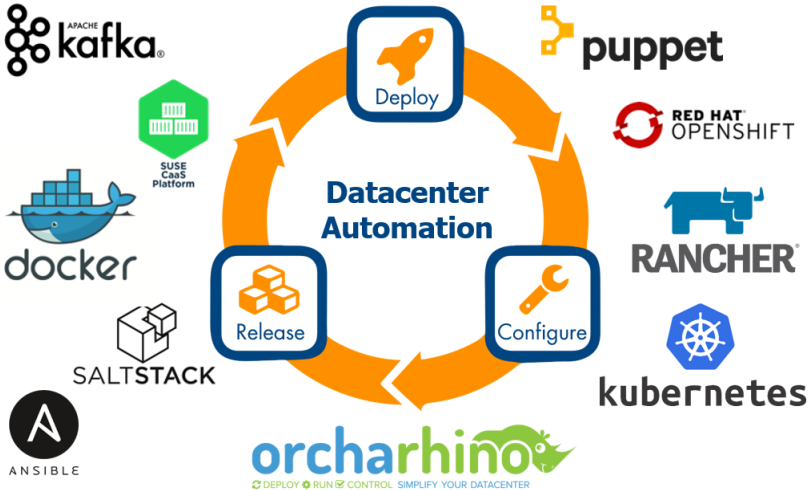
Use Cases

- Event Delivery/Streaming
- Website Activity Tracking
- Monitoring
- IoT-Messaging
- Storage

Read more

- ATIX Crew - https://www.heise.de/select/ix/2019/4/1553935521978043
- Lukas Berle [1] - https://www.informatik-aktuell.de/betrieb/verfuegbarkeit/apache-kafka-eine-schluesselplattform-fuer-hochskalierbare-systeme.html
- https://kafka.apache.org/
- https://docs.confluent.io/current/
- Google ;-)

Kafka is only part of the story!



curl -X GET localhost://whoami/contactme

```
{"me":
  {
    "xing/linkedin":
    "Bernhard Hopfenmüller",
    "email":
    "hopfenmueller@atix.de",
    "IRC/Github/Twitter":
    "Fobhep",
  }
}
```