Apache Kafka...

..."a system optimized for writing"

Bernhard Hopfenmüller

23. Oktober 2018

whoami

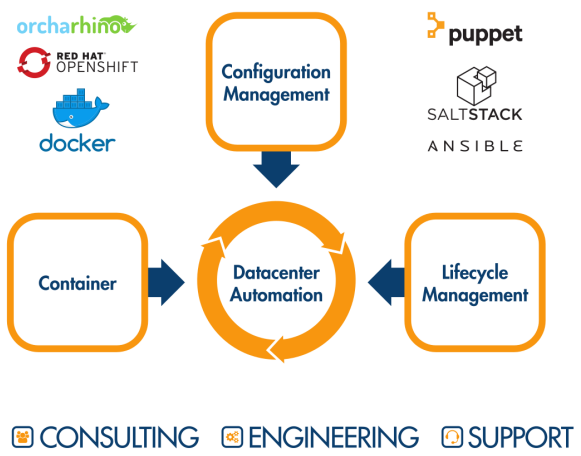
Bernhard Hopfenmüller
IT Consultant @ ATIX AG

IRC: Fobhep
github.com/Fobhep

whoarewe

The Linux & Open Source Company
Unterschleißheim @ München

over 15 years
datacenter automation, Linux
Consulting, Engineering, Support,
Training



Kafka

Quora.com

What is the relation between Kafka, the writer, and Apache Kafka, the distributed messaging 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 colleague and liked Franz Kafka. Plus the name sounded cool for an OS project

- developed by LinkedIn, Open Source since 2011
- 2014 foundation of Confluent 

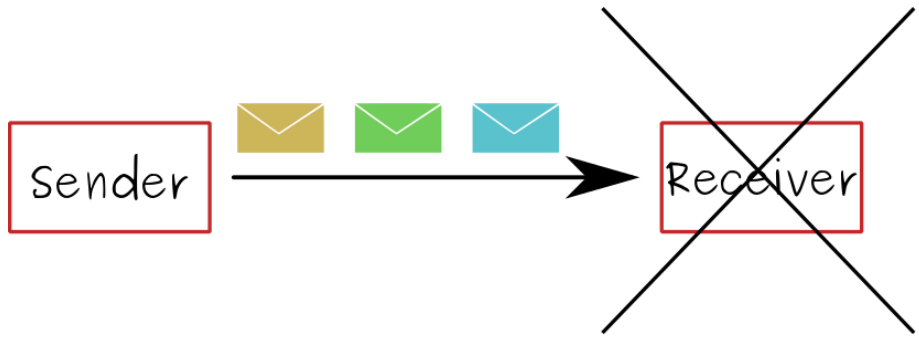
Messaging-Systems

Why do we need a messaging system?

Messaging-Systems

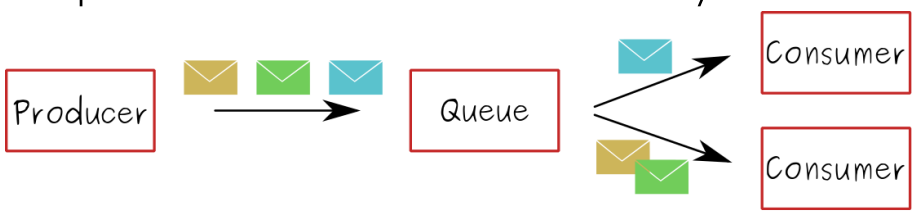
Why do we need a messaging system?

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

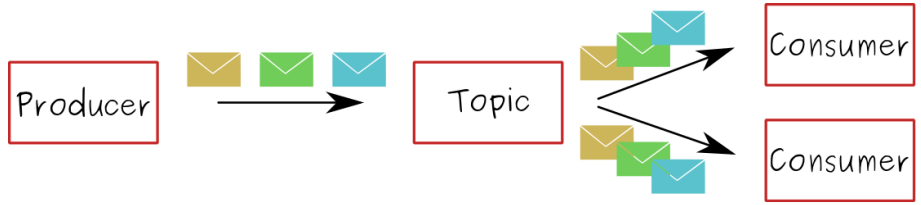


Supermarket vs Television Source[1]

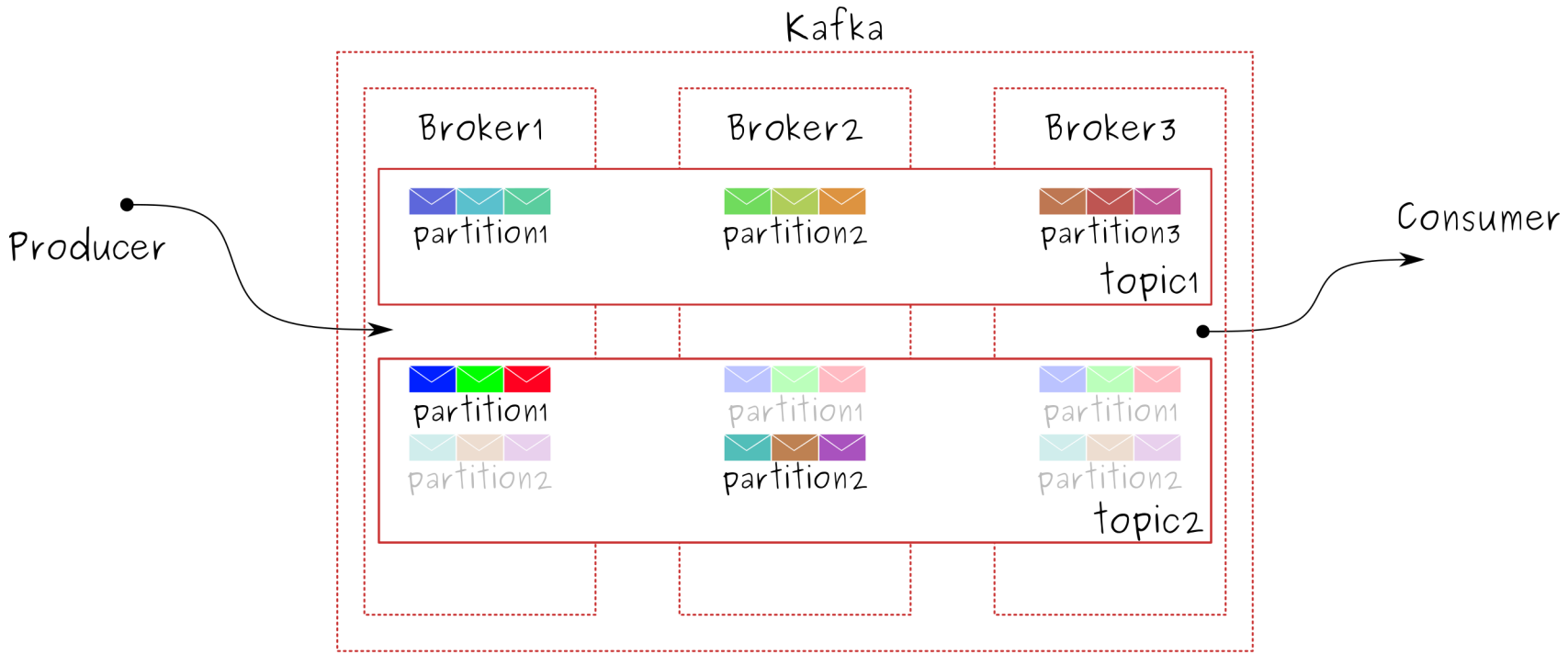
Supermarket Wait until it's your turn



Television Choose what you want to receive



Kafka-Basic structure

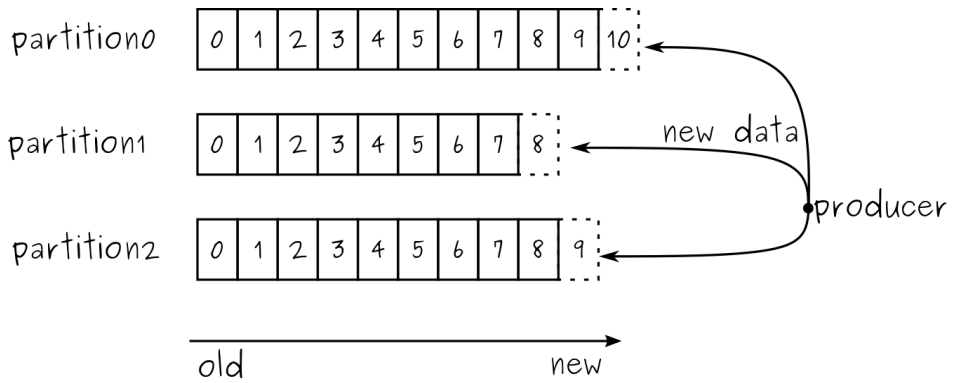


Use Cases

- Messaging (ActiveMQ or RabbitMQ)
- Website Activity Tracking
- Metrics
- Log Aggregation
- Stream Processing
- Apache Storm and Apache Samza.
- Commit Log

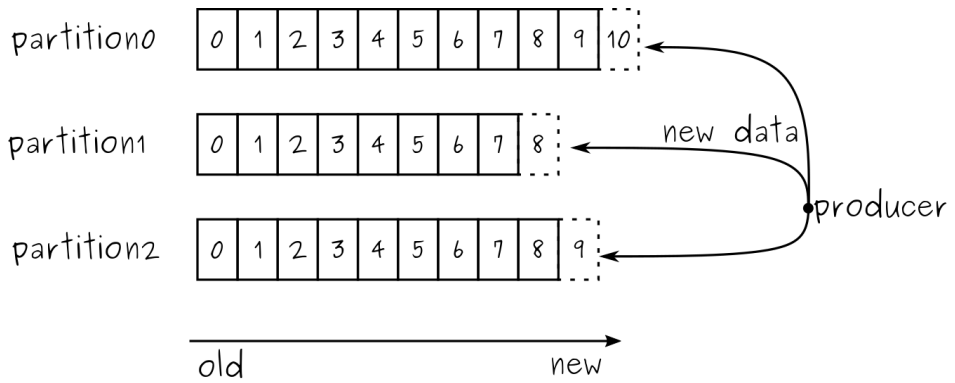
Topics I

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

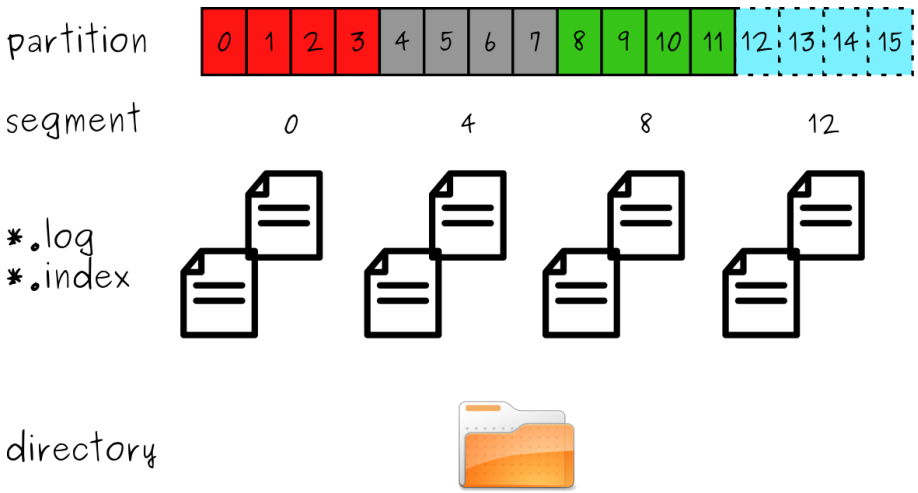


Topics II

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

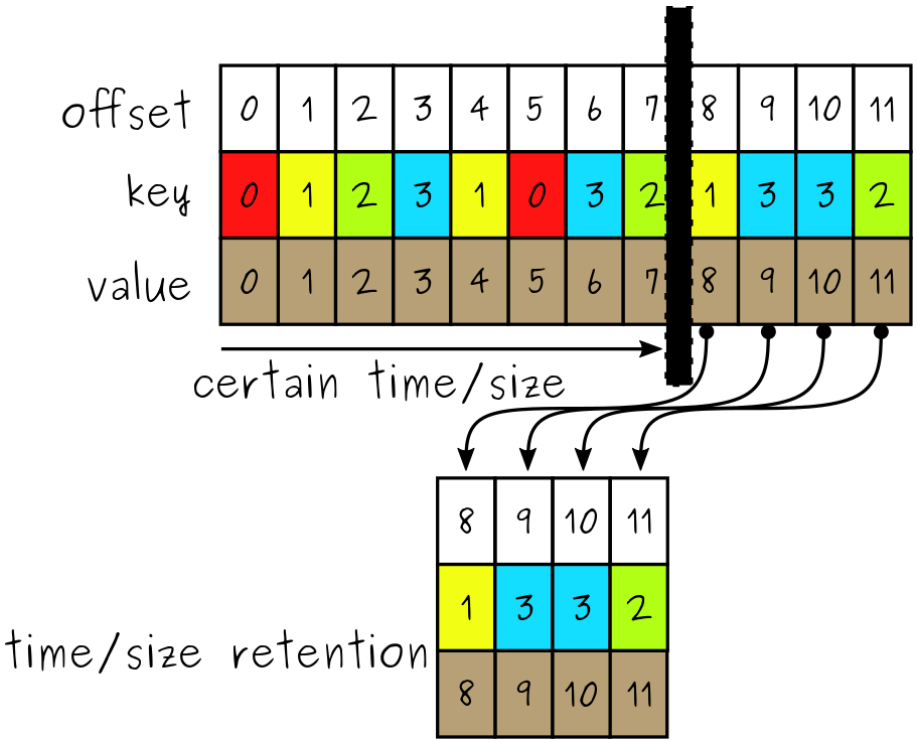


Topics III



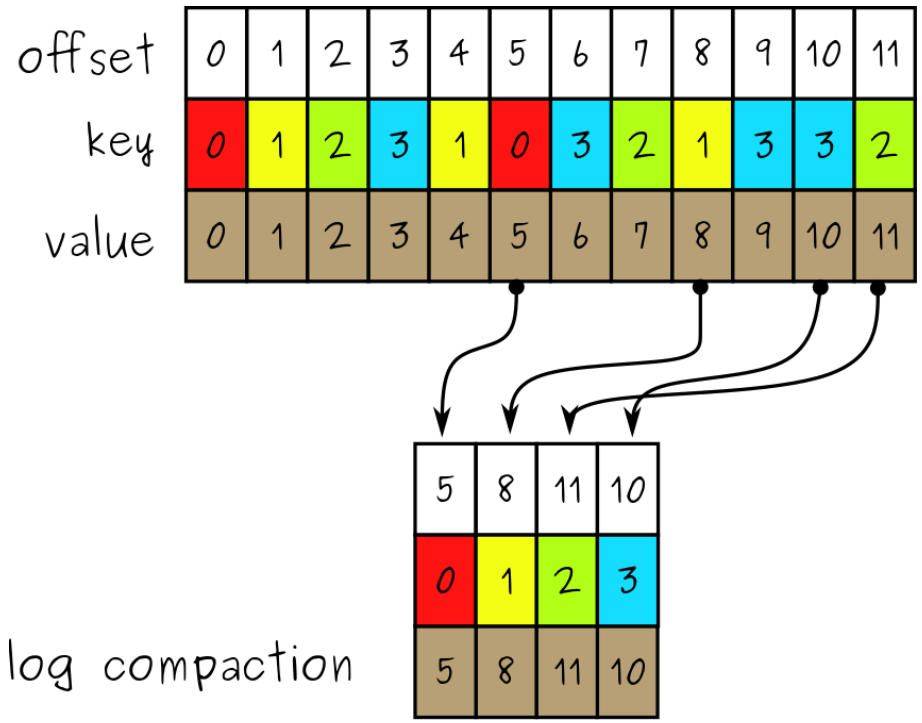
- messages are stored physically!
- key-value principle
- Clean-Up policies:

Topics IV



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

Topics V



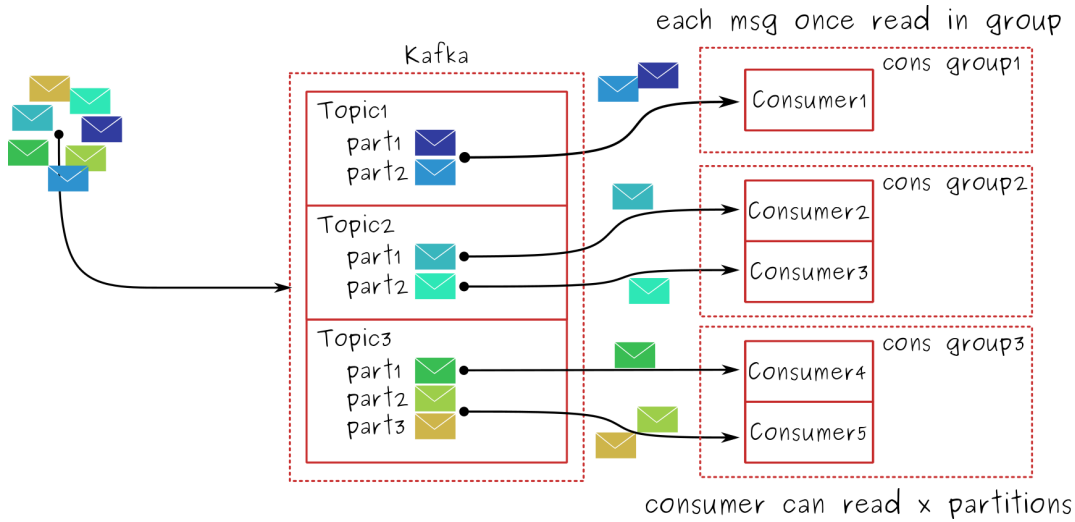
- Clean-Up policies:
  - default: Retention-time (delete old data after *x* days)
  - Retention-size (delete old data if data memory > x)
  - Log-Compaction (replace old value to key with new)

Topic consumption

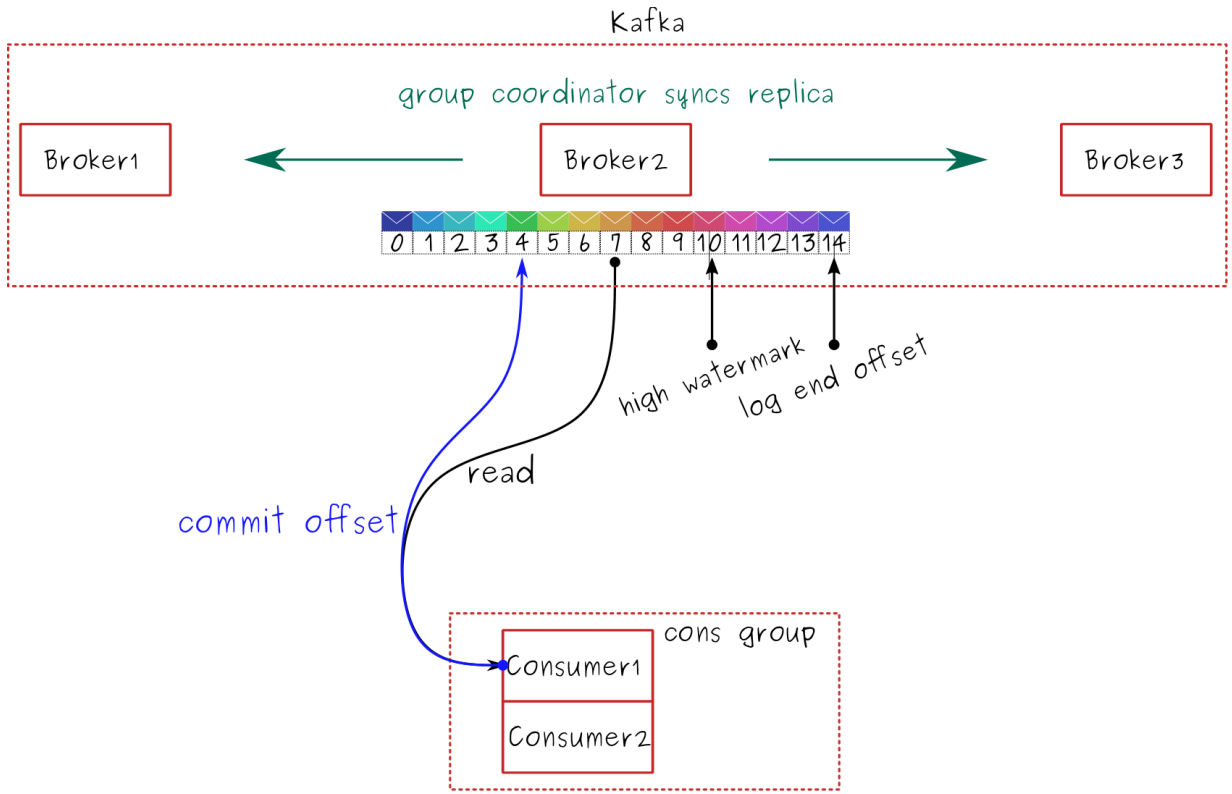
- topics are pulled! (no DoS)
- any existing data can be pulled

Consumer Groups

- parallelism allows high throughput
- never more consumers than partitions
- Kafka features exactly-once-semantics!



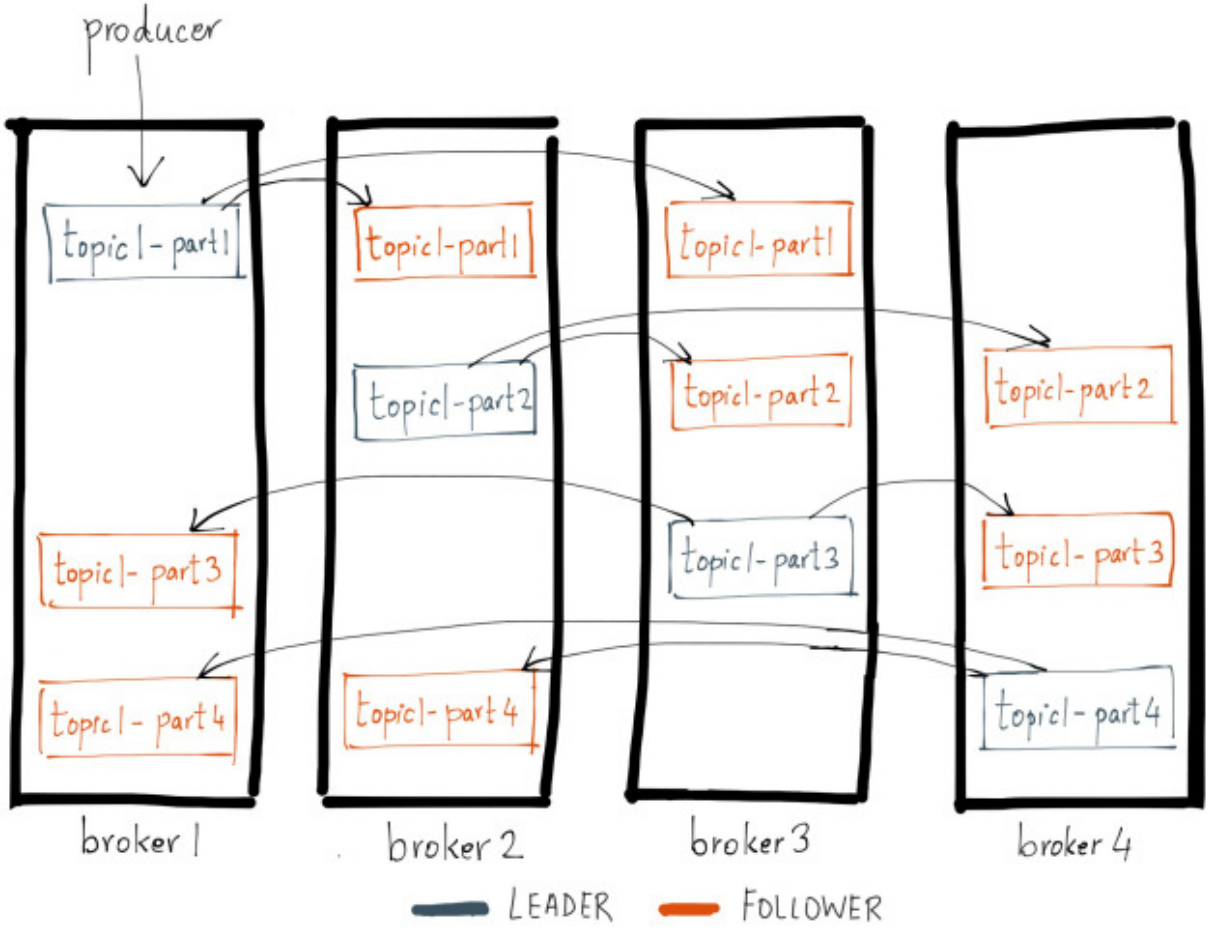
Wait but who knows what's read?



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

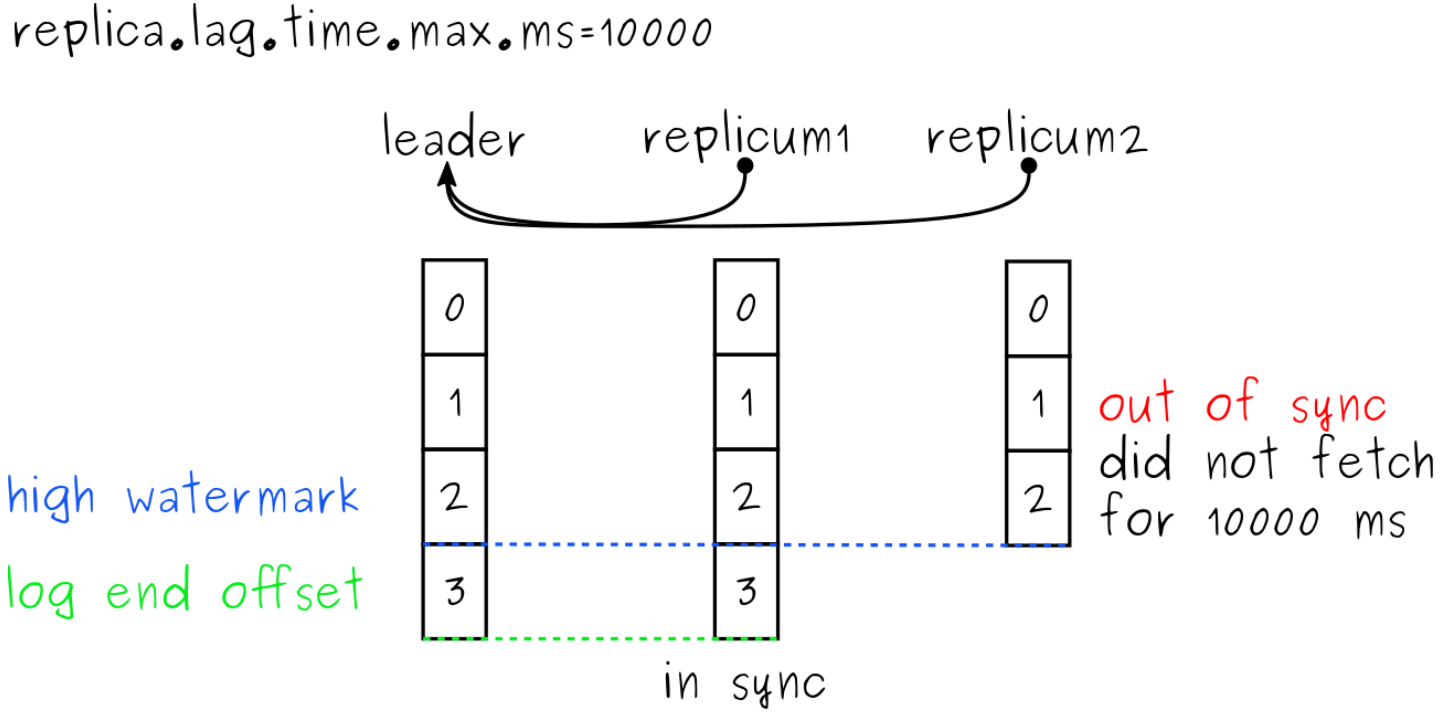
Replication

implemented on partition level



Source[3]

In and Out of Sync Replica



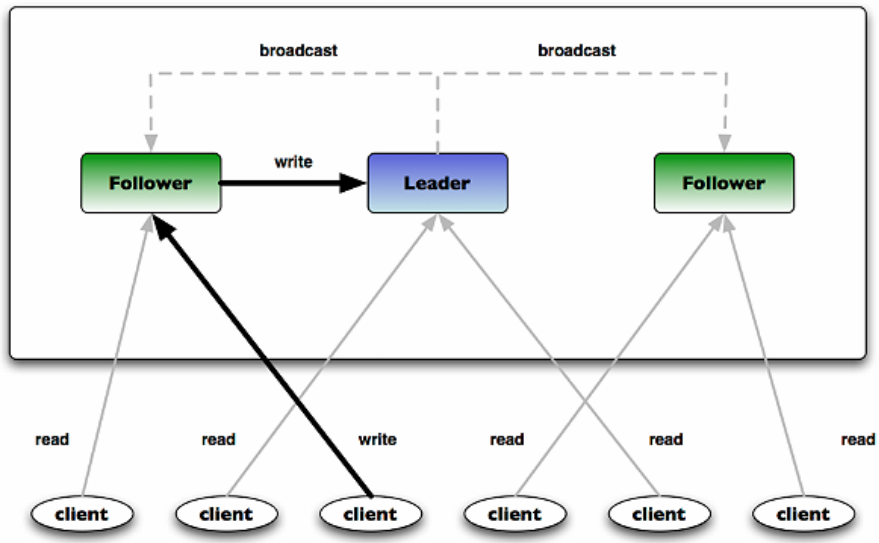
Did somebody hear my message?

Producer decides if message was successfully sent
Configuration possibilities:

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

ZooKeeper

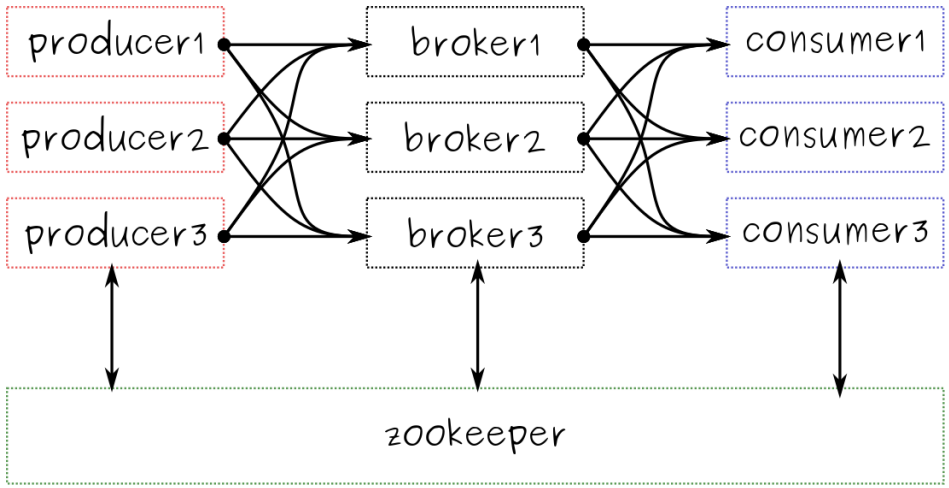
- distributed, hierachical file system
- management of znodes()
- HA via ensemble (=ZooKeeper cluster)



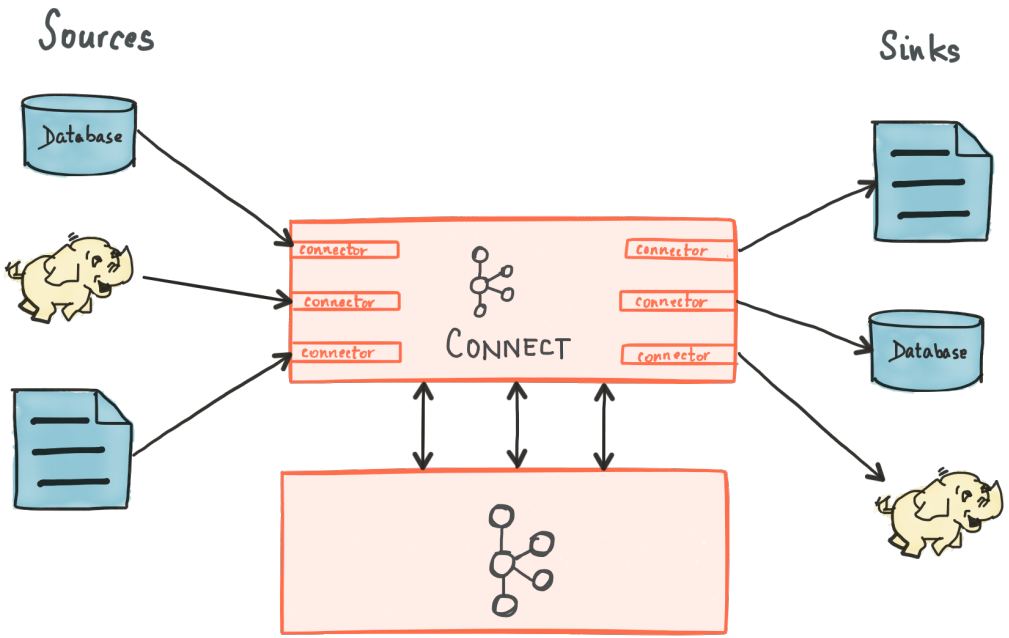
Source[4]

Broker and ZooKeeper

- Brokers are stateless!
- Which Broker is alive?
- Broker communication?
- → ZooKeeper!



Talk to Kafka - Kafka Connect

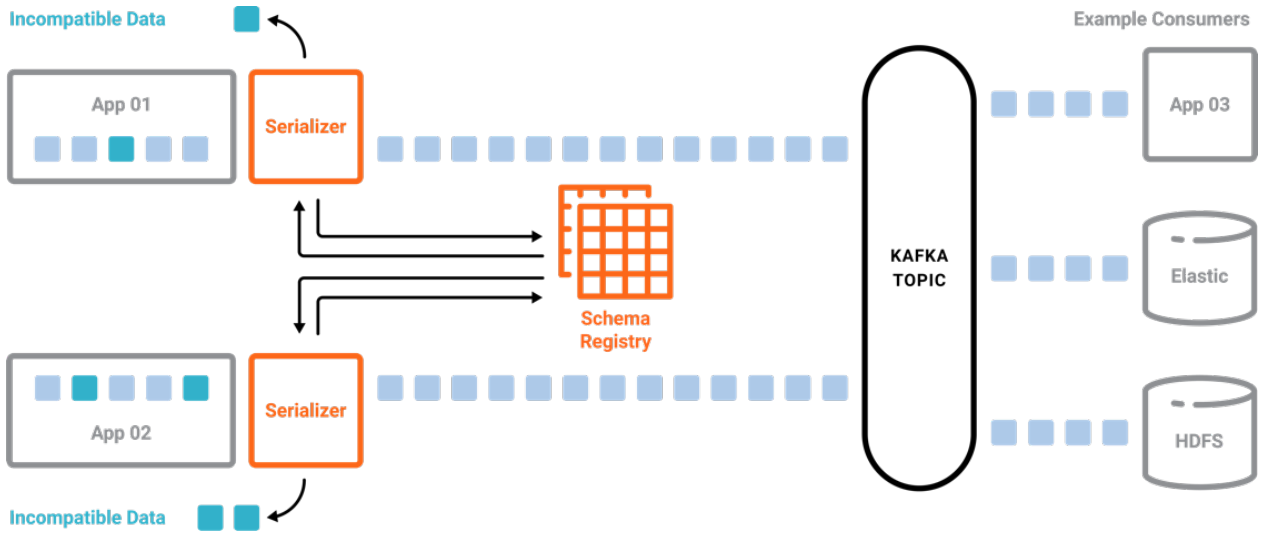


- I/O for Kakfa
- Connect with external systems
- Open Source by Confluent

Source[7]

Talk to Kafka - Schema Registry

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



source: confluent

TV or Netflix?

Hello, Streaming World

KSQL

```
CREATE STREAM fraudulent_payments AS
 SELECT * FROM payments
 WHERE fraudProbability > 0.8;
```

You write ***only*** SQL. No Java, Python, or other boilerplate to wrap around it!

But you can create KSQL User Defined Functions in Java, if you want to.

kafka streams

```
object FraudFilteringApplication extends App {

  val config = new java.util.Properties
  config.put(StreamsConfig.APPLICATION_ID_CONFIG, "fraud-filtering-app")
  config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092,kafka-broker2:9092")

  val builder: StreamsBuilder = new StreamsBuilder()
  val fraudulentPayments: KStream[String, Payment] = builder
    .stream[String, Payment]("payments-kafka-topic")
    .filter((_ ,payment) => payment.fraudProbability > 0.8)

  val streams: KafkaStreams = new KafkaStreams(builder.build(), config)
  streams.start()
}
```

source: confluent

- live filtering of topics
- KSQL!
- Open Source by Confluent

Who likes Kafka?

- zalando - microservices
- Cisco Systems - security
- Airbnb - event pipeline
- Netflix (Monitoring!)
- The New York Times (Kafka as data storage! Super awesome blog post) [5][6]
- Audi - IoT
- Spotify
- Twitter
- Uber (Kafka = Backbone!!!)
- https://kafka.apache.org/powered-by

Sources

1 https://www.informatik-aktuell.de/betrieb/verfuegbarkeit/apache-kafka-eine-schluesselplattform-fuer-hochskalierbare-systeme.html

2 https://thecattlecrew.net/2017/09/28/apache-kafka-im-detail-teil-1/ and https://thecattlecrew.net/2017/09/28/apache-kafka-im-detail-teil-2/

3 https://www.confluent.io/blog/hands-free-kafka-replication-a-lesson-in-operational-simplicity/

4 https://www.infoq.com/articles/apache-kafka

5 https://www.confluent.io/blog/okay-store-data-apache-kafka/

6 https://www.confluent.io/blog/publishing-apache-kafka-new-york-times/

7 https://www.confluent.io/blog/simplest-useful-kafka-connect-data-pipeline-world-thereabouts-part-1/

Install Kafka with Docker/Ansible

- Run containers as services
- No SSL/SASL yet!
- have a look at playbooks and docker-compose files
- https://github.com/confluentinc/cp-ansible
- https://docs.confluent.io/current/installation/docker/docs/installation/index.html
- Wurstmeister: https://github.com/wurstmeister/kafka-docker

Single Components

```
---
- name: Start zookeeper
  docker_container:
    name: zookeeper
    image: "{{ images.zookeeper }}:{{ versions.kafka }}"
    state: started
    restart_policy: unless-stopped
    ports:
     - "{{ ports.zookeeper.client }}:2181"
     - "{{ ports.zookeeper.peer }}:2888"
     - "{{ ports.zookeeper.leader }}:2181"
    volumes:
      - "/zookeeper/data:/var/lib/zookeeper/data"
      - "/zookeeper/log:/var/lib/zookeeper/log"
    env:
      ZOOKEEPER_SERVER_ID: "{{ zookeeper_server_id }}"
      ZOOKEEPER_CLIENT_PORT: "2181"
      ZOOKEEPER_SERVERS: "{{ lookup('template', 'sort_zookeeper.j2') }}"
      ZOOKEEPER_DATA_DIR: "/var/lib/zookeeper/data"
      ZOOKEEPER_LOG_DIR: "/var/lib/zookeeper/log"
```

```
{% for host in groups['zookeeper'] %}
  {% if inventory_hostname == hostvars[host]['inventory_hostna
    0.0.0.0
  {% else %}
    {{ hostvars[host]['ansible_default_ipv4']['address'] }}
  {% endif %}
  {% if not index_loop.last %}
    ;
  {% endif %}
{% endfor %}
```

Check system health

```
---
- name : "Check Zookeeper Health"
  command : docker run --rm -it confluentinc/zookeeper cub zk-re
  register : output
  until: output is success
  retries: 3
...
```

Configure via REST/uri

```
---
- name: create new topic
  command: "{{ 'sudo docker run --rm confluentinc/cp-kafka
                kafka-topics --create' ... }}"

- name: get information of current topic
  uri:
    url: "{{ restproxy_url ~ /topics/' + topic.name }}"
    register: result

...
```

whoami

Bernhard Hopfenmüller

IRC: Fobhep
github.com/Fobhep twitter.com/fobhep

Kafka vs MQ

- Kafka has no P2P model!
- Messages are Persistent!
- Topic Partitioning!
- Message Sequencing: for one partition (send order=received order)
- Message reading: Choose where to read, Rewind, no FIFO!
- Loadbalancing: automatic distribution easier with metadata
- HA and failover implemented very easily