Zero to Kafka in 10 Seconds

Laura Uzcategui
Software Engineer @ Workday
@laura_uzcategui

Agenda

Introduction

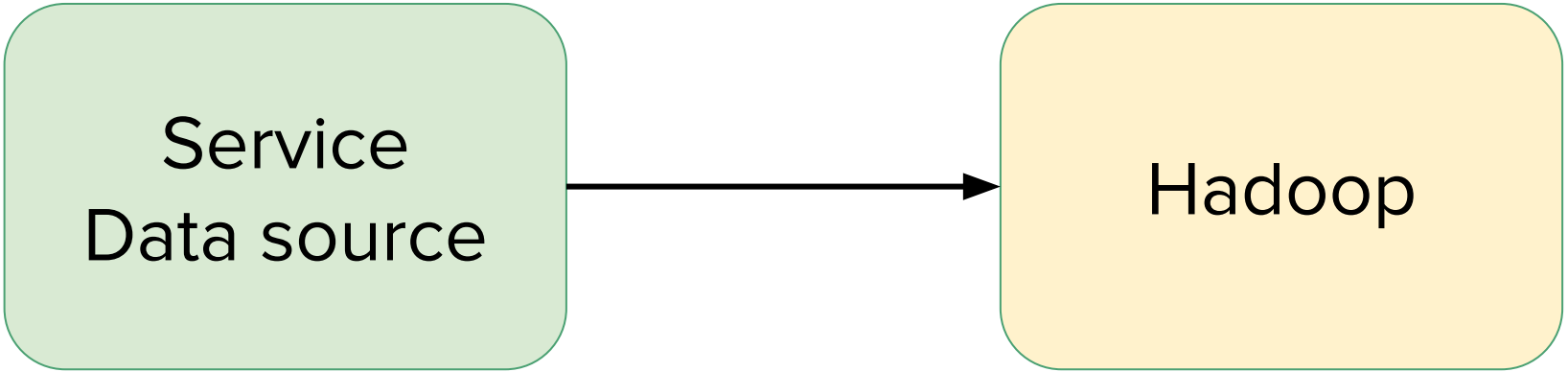
What is a log ?

Kafka as a Messaging System

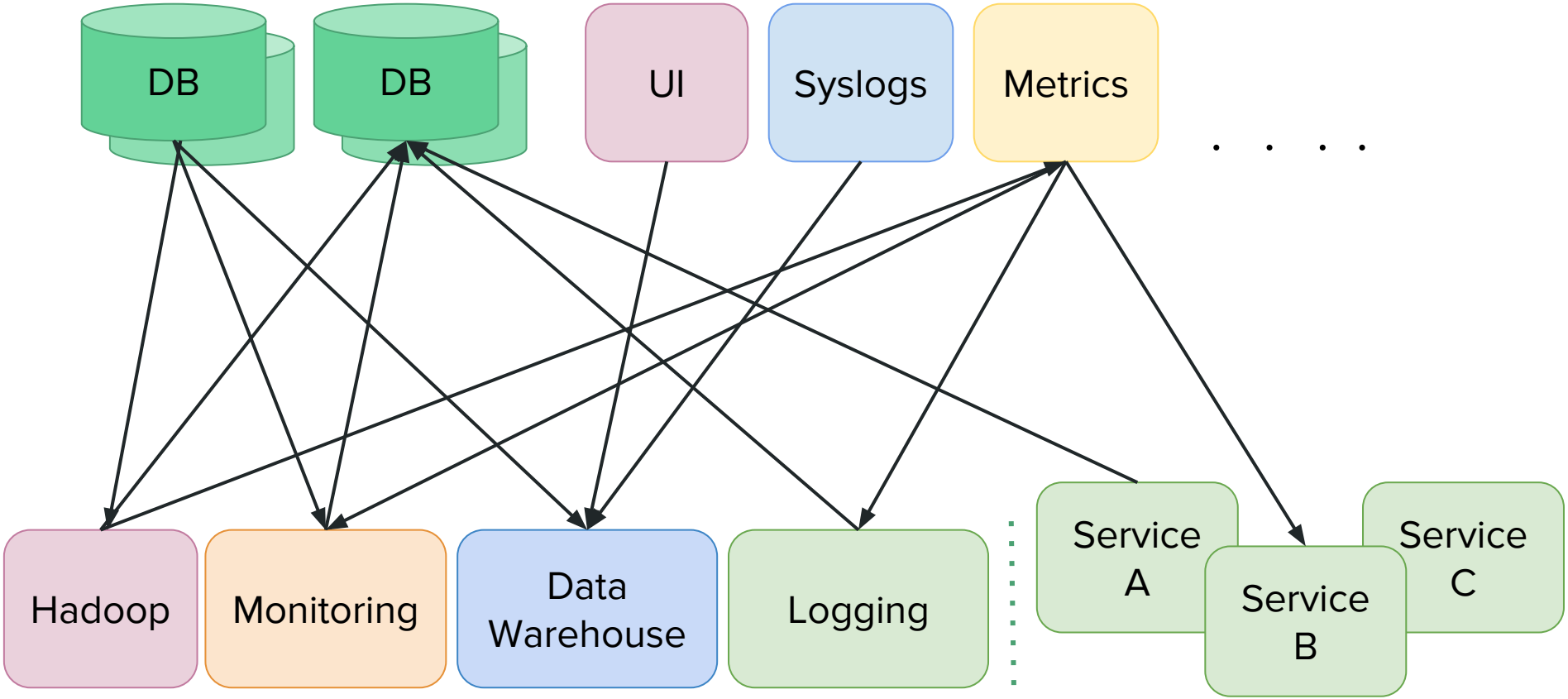
Kafka Components

Stand up a Kafka Cluster in 10 seconds with Docker

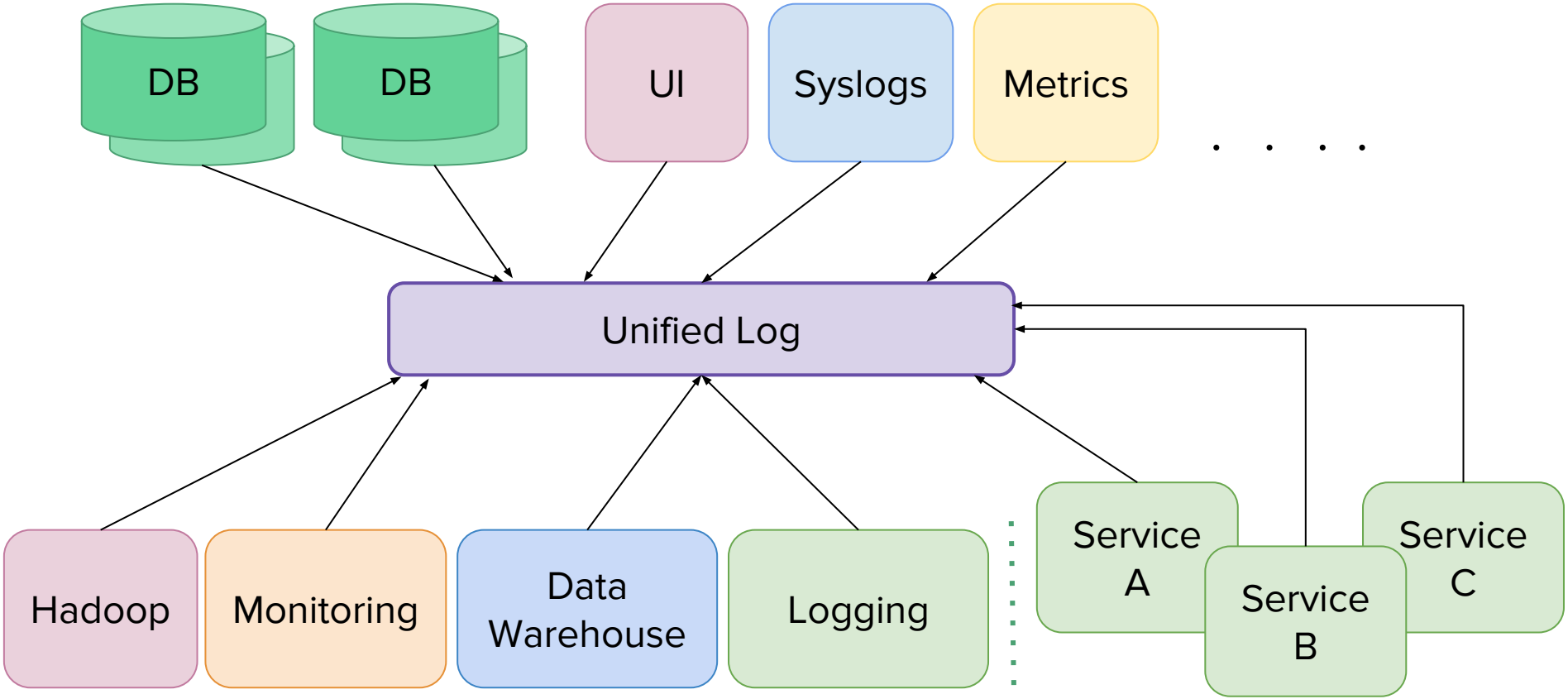
Ideal World



Real World


DB
DB
UI
Syslogs
Metrics
Hadoop
Monitoring
Data Warehouse
Logging
Service A
Service B
Service C

Unified Log Processing



What is a Log?

```
[laura.uzcategui] ~ $ tail -f -n 20 /var/log/system.log
Jul 26 18:04:14 C02QV089G8WL galileod[58201]: BGSync: Client Has No Connected Transports
Jul 26 18:04:14 C02QV089G8WL galileod[58201]: BGSync: Background Sync | End
Jul 26 18:05:34 C02QV089G8WL syslogd[42]: ASL Sender Statistics
Jul 26 18:05:54 C02QV089G8WL com.apple.xpc.launchd[1] (com.apple.xpc.launchd.user.domain.
ed to perform action: plugin-containe.69611, action = pid-local registration, code = 1: 0
 euid = 1493760548, gid = 1933861586, egid = 1933861586, asid = 100006
Jul 26 18:06:24 --- last message repeated 1 time ---
Jul 26 18:07:14 C02QV089G8WL galileod[58201]: BGSync: Background Sync | Start
Jul 26 18:07:14 C02QV089G8WL galileod[58201]: BGSync: Client Has No Connected Transports
Jul 26 18:07:14 C02QV089G8WL galileod[58201]: BGSync: Background Sync | End
Jul 26 18:08:29 C02QV089G8WL SystemUIServer[913]: notify name "user.uid.1493760548.com.ap
nTouchBarChanged" has been registered 540 times - this may be a leak
Jul 26 18:08:31 C02QV089G8WL login[69619]: USER_PROCESS: 69619 ttys004
Jul 26 18:10:14 C02QV089G8WL galileod[58201]: BGSync: Background Sync | Start
Jul 26 18:10:14 C02QV089G8WL galileod[58201]: BGSync: Client Has No Connected Transports
Jul 26 18:10:14 C02QV089G8WL galileod[58201]: BGSync: Background Sync | End
```

What is a Log?

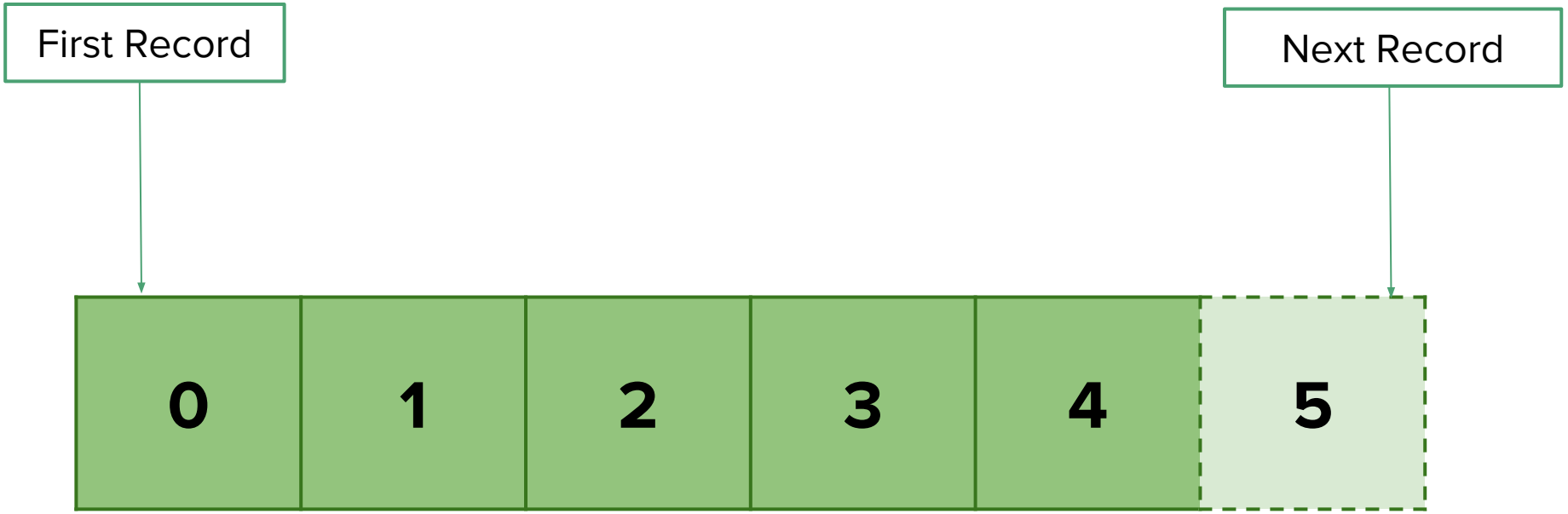
It could be seen as an unstructured pieces of data on the that helps you to trace, track any transaction/event that are happening within your system and they are usually Readable

BUT.....

What is a Log ?

In the DB world, We would like to keep a record of the transactions that are pending (Journals)

- *Sequential Ordering*
- *Reads from Left to Right*
- *Append Only*


First Record
Next Record
0
1
2
3
4
5


kafka

“Publish/subscribe messaging system”

“A distributed commit log”

“Distributing streaming platform”

What Kafka is not for

- Synchronous communication
- Messages delivery to a particular location, i.e server X
- Long term storage

Kafka Components & Characteristics

- Producer
- Schemas
- Topic
- Partition
- Offset

- Broker
- Leader Election

- Consumer
- Consumer Group

Let’s do an Analogy first :)

Producer

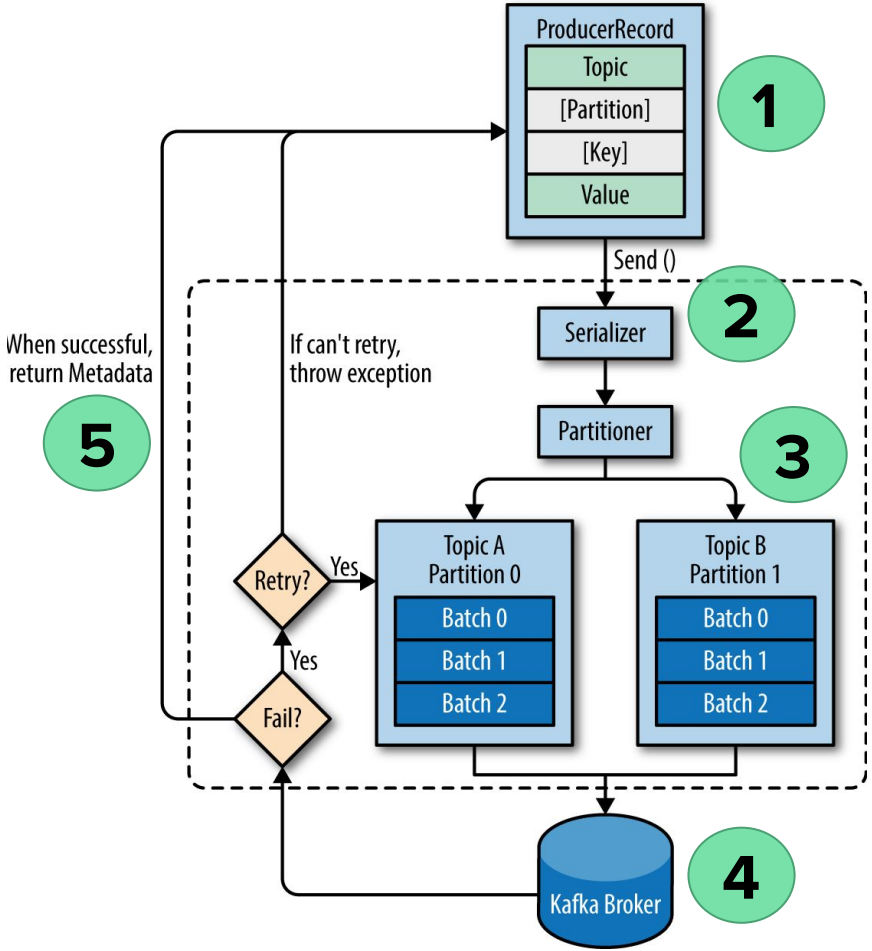
Producer

- Create new messages (possibly with a schema)
- Publish messages to a Topic
- It could address the message to a specific partition.

Depending on your use case, requirements might be different:

  - Latency
  - Duplicated messages
  - Throughput

Kafka Producer Components



Producer Configuration

- **ack** : how many acknowledgments must be received from the leader before consider a **Successful Write**
- **buffer.memory:** amount of memory for buffering messages before sending to the broker
- **retries**: how many times the producer will retry to send a message
- **Max.request.size:** it caps the size of the larger message and the maximum amount of messages

Spoiler Alert: significant impact on memory use, performance, and reliability

Schemas - Serialization

Using serializers and making use of schema definition make it easier for having a:

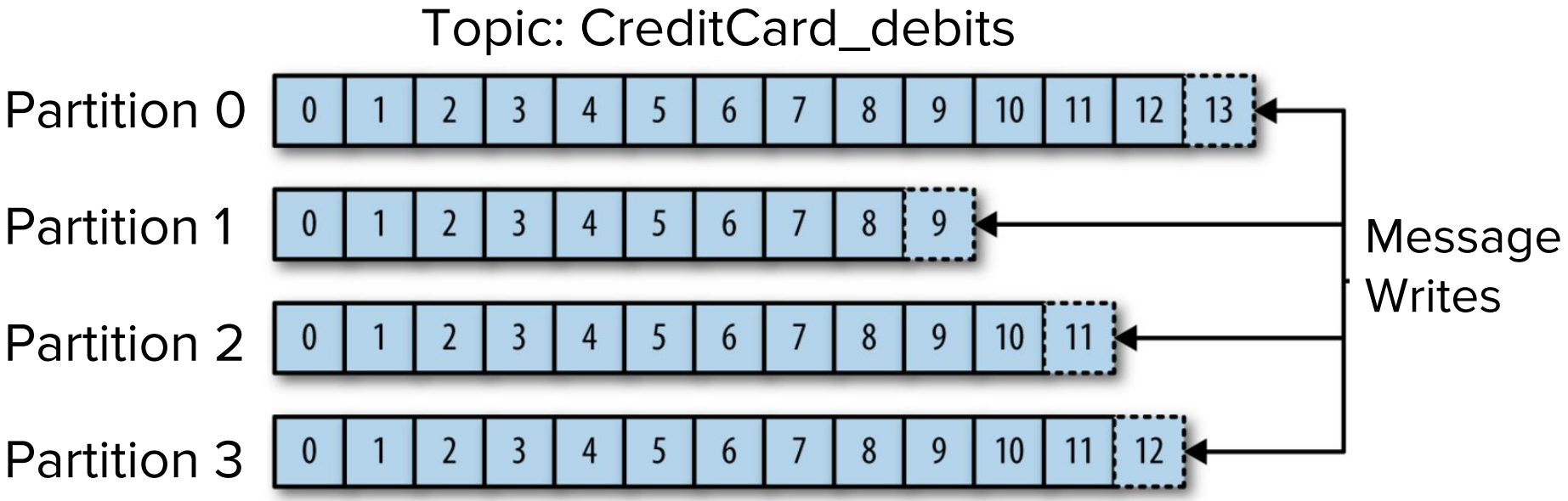
- Common understanding of the records
- Versioning
- Data dictionary - Schema Registry
- Backwards / Forward Compatibility

Serialization Libraries

Avro, Protobuf, Thrift, CapNProto, Json and more ...

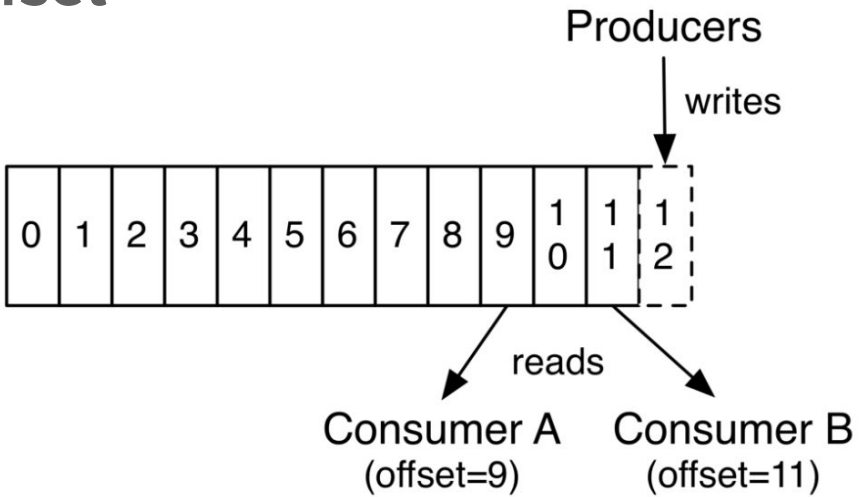
Topic

“ It is a Stream of messages of a particular type”

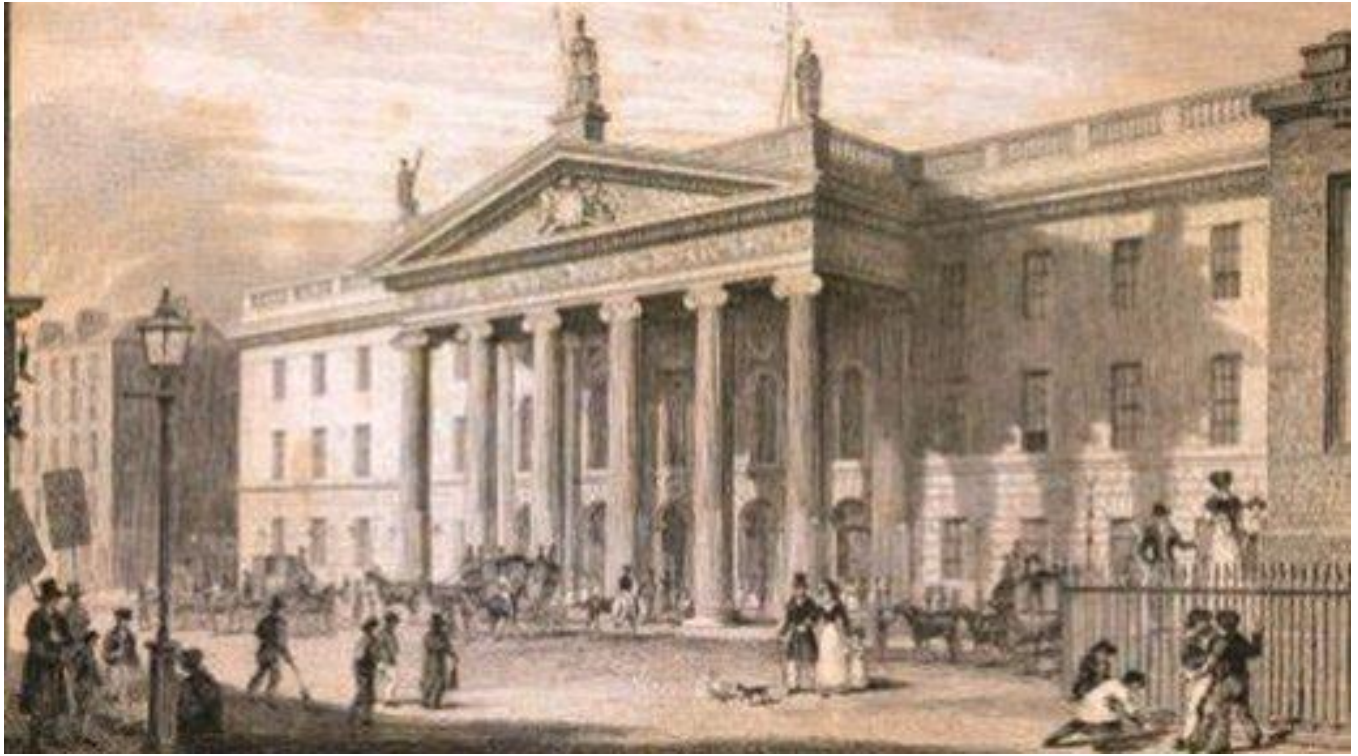


Partition & Offset

- Each partition is a log (commit-log)
- Partitions are distributed and replicated among brokers
- Records inside are identified by sequential id number called the **offset**



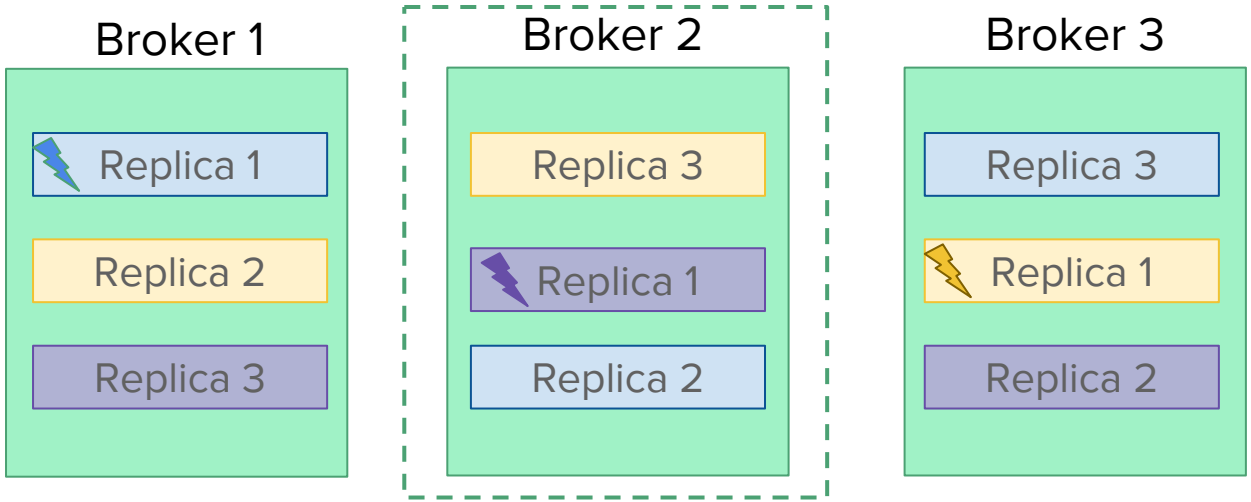
Broker



Broker

“ Published messages are stored on servers called Brokers”

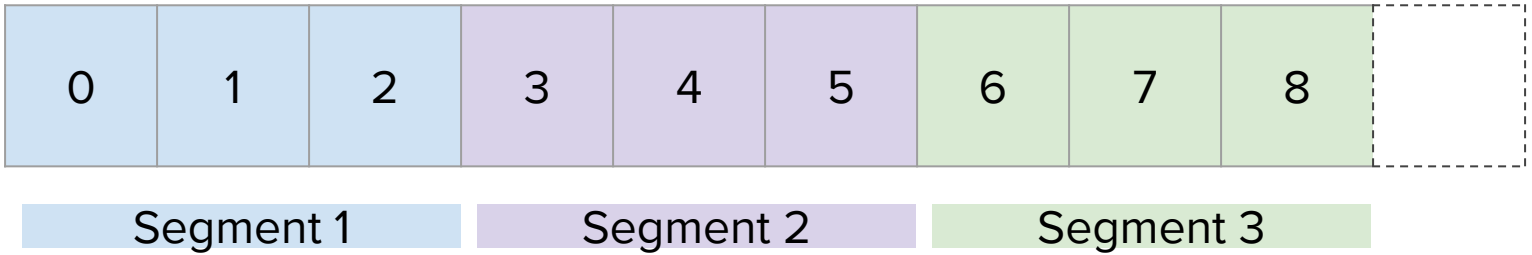
- Standalone or Cluster
- Each broker holds a leader and replicas of a partition
- One of the brokers functions as Controller



Broker

- Store all messages on disk.
- Brokers are stateless
- Messages has a retention period on the broker

Physical Storage of a Partition (Log)



Broker - Physical Storage

```
bash-4.4# tree -C /kafka/kafka-logs-787134c8e7ec/twitter_handlers-*/
/kafka/kafka-logs-787134c8e7ec/twitter_handlers-0/
├── 00000000000000000000.index
├── 00000000000000000000.log
├── 00000000000000000000.timeindex
└── leader-epoch-checkpoint
/kafka/kafka-logs-787134c8e7ec/twitter_handlers-1/
├── 00000000000000000000.index
├── 00000000000000000000.log
├── 00000000000000000000.timeindex
└── leader-epoch-checkpoint
```

Replication

- It’s the <3 of Kafka Architecture.
- Guarantees availability and durability of the logs.

Type of Replicas

Leader

- All requests to produce/consume arrives to the leader
- Keeps record of followers being in sync
- When partitions are created there is a preferred leader

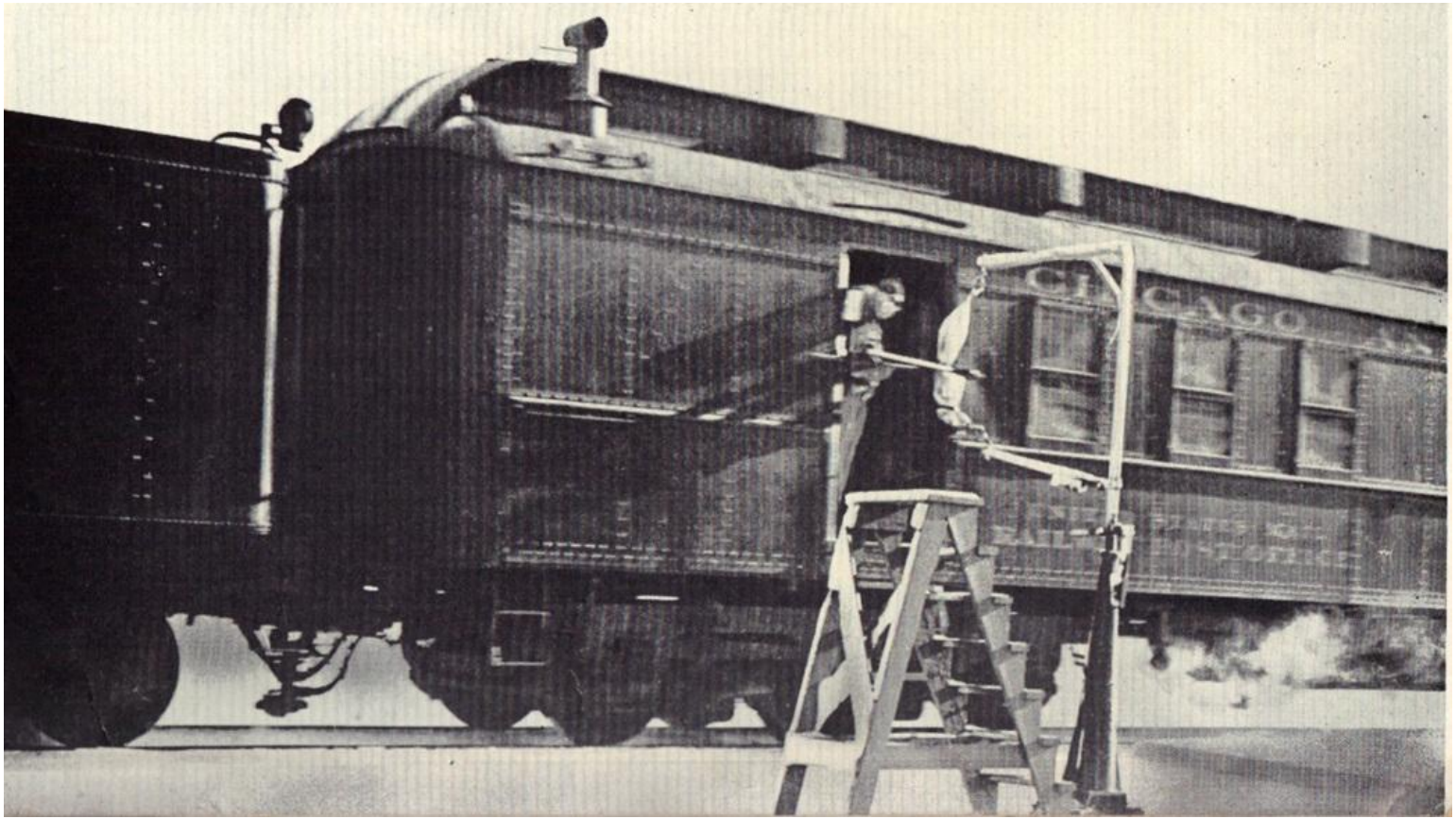
Follower

- Only job is replicate messages from leader
- If the leader crash, one of the followers get elected Leader
- Only in-sync replicas can be leaders

Leader Election

- Kafka uses Zookeeper to do leader election of Broker and topic partitions.
- It doesn’t do it by majority vote (consensus algorithms)
- Instead it uses the In-Sync-Replica sets
- Only members who are ISR are able to be elected as Leaders
- The controller is responsible for change the leader of affected partitions on a failed broker.

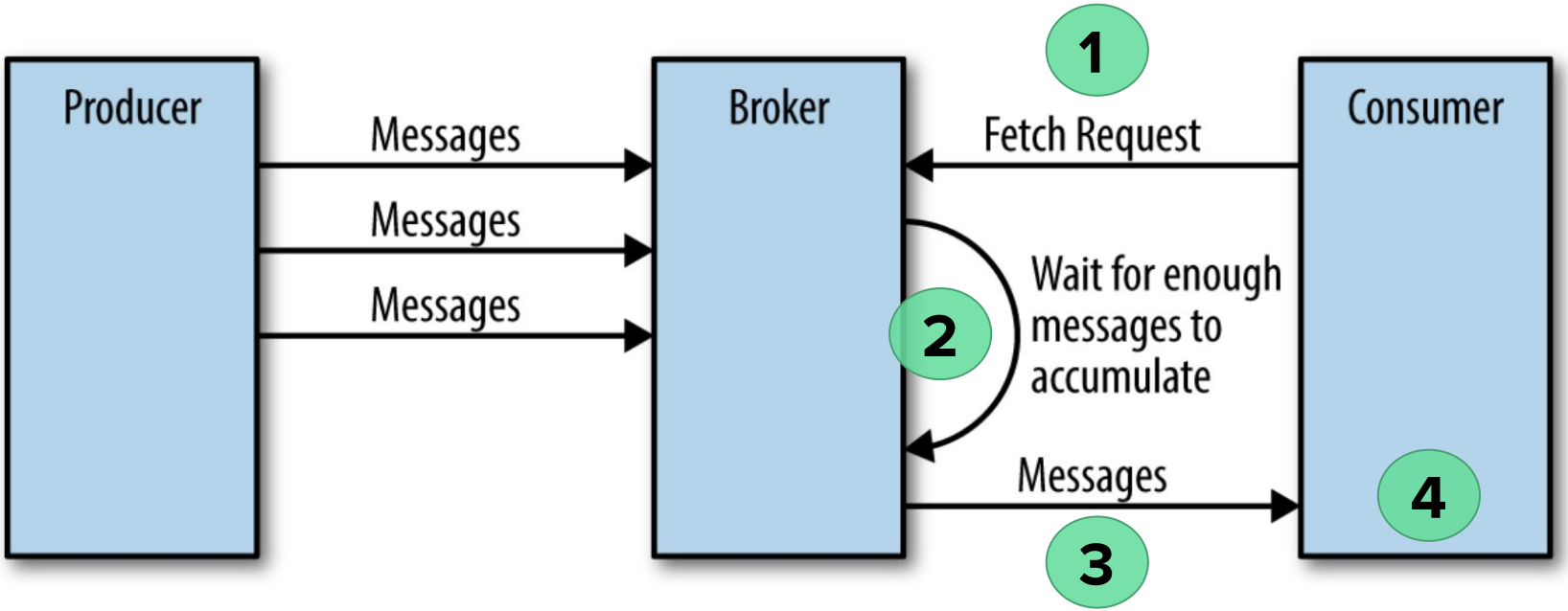
Consumer & Consumer Groups



Consumer

- Consume messages from a single partition
- It works with subscription against 1+ topic
- It use a pull request method against the brokers
- Keeps a record of the offset that it reads from
- Responsible for the offset commit
- Only see messages that were already replicated
- It attach itself to a Consumer Group

Consumer Pull Request

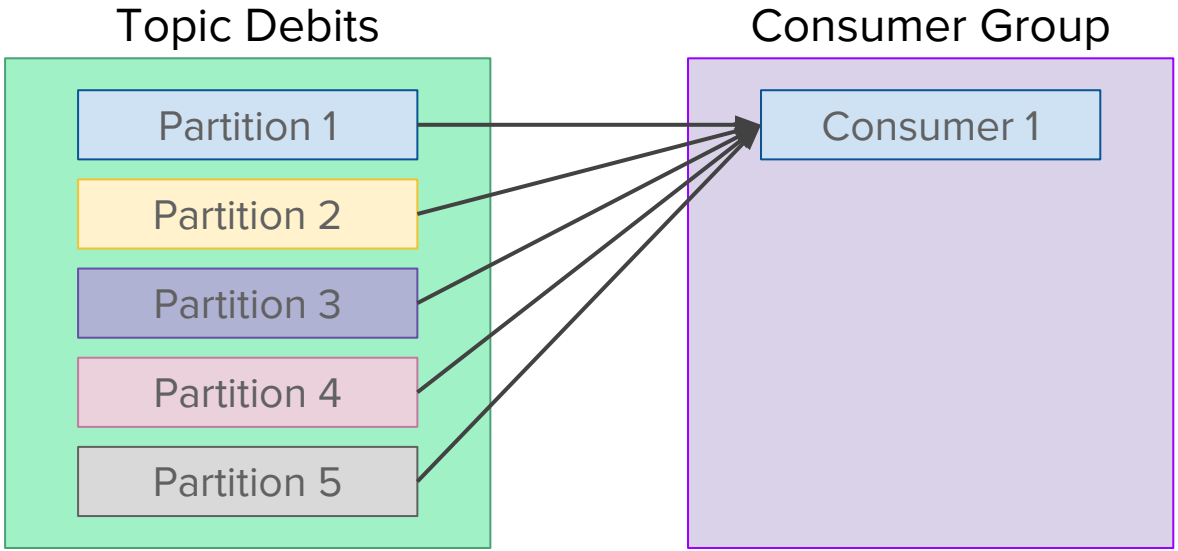


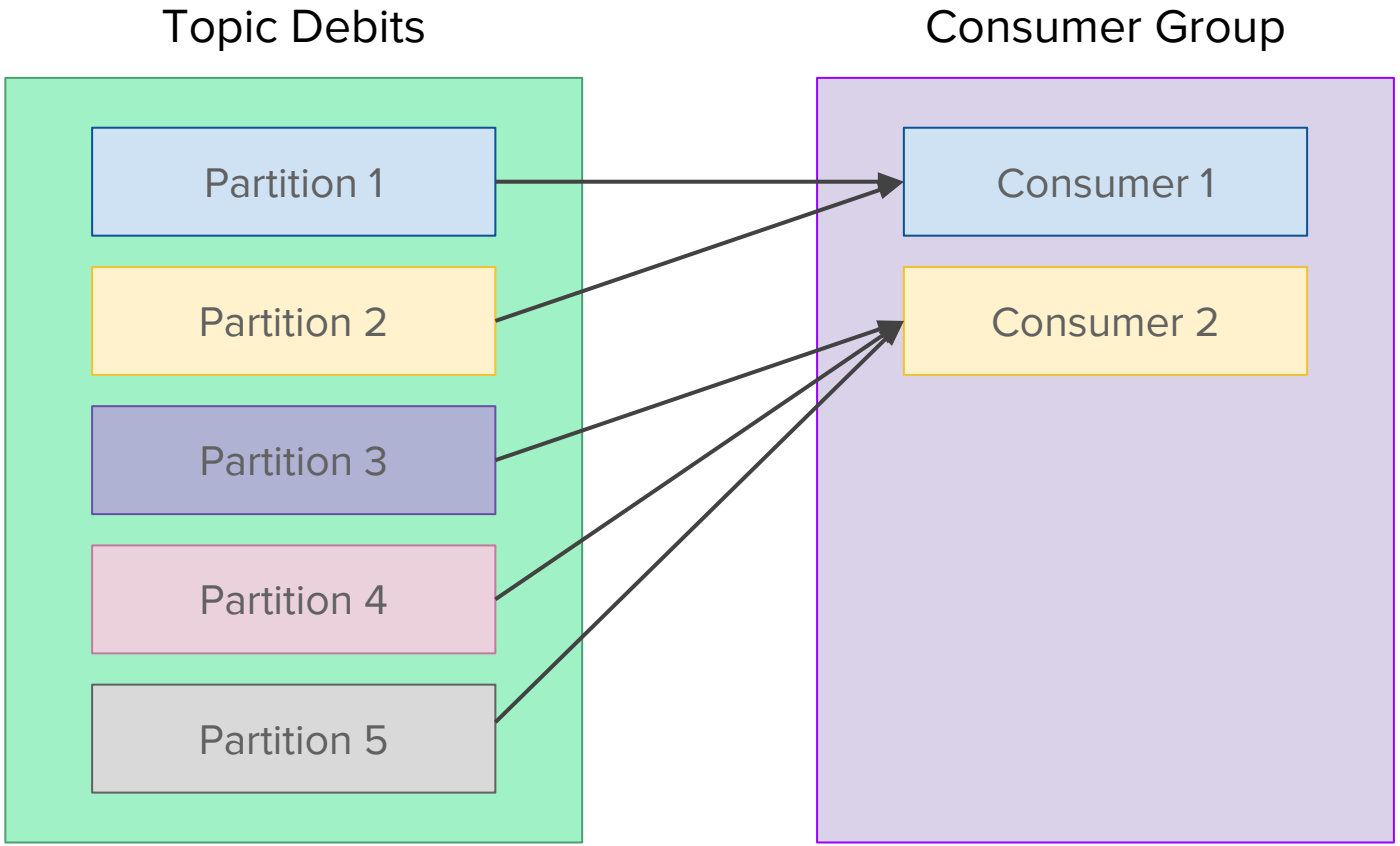
Consumer Configuration

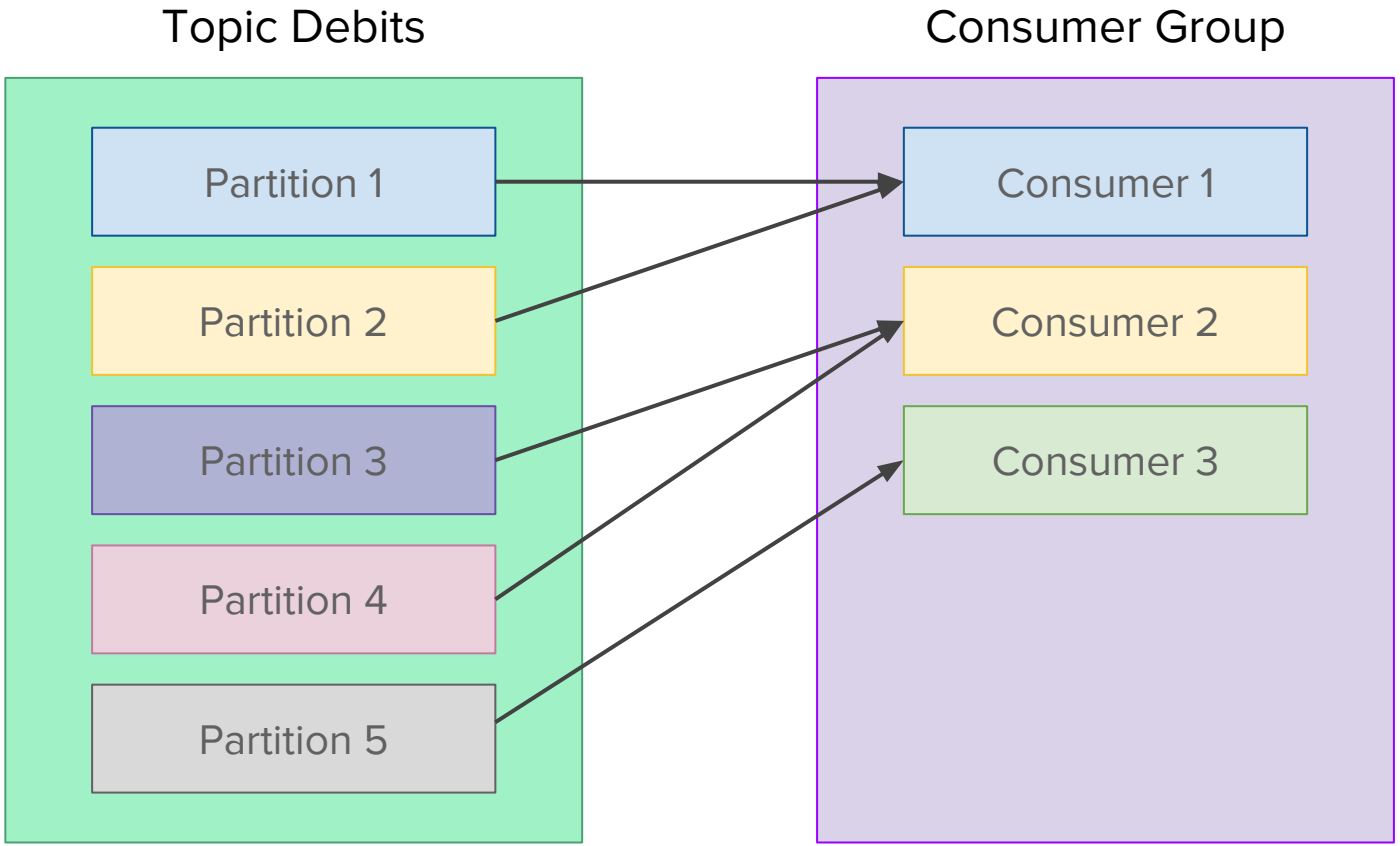
- **group.id**: name of consumer group it belongs
- **fetch.min.bytes**: minimum amount of data to fetch
- **fetch.max.wait.ms**: wait period until there is enough data. (timeout)
- **max.partition.fetch.bytes**: max amount of bytes returned by partition
- **auto.offset.reset:** where in the partition start reading from
- **Partition.assignment.strategy:** strategy for partition assignment (round robin , range or your own)

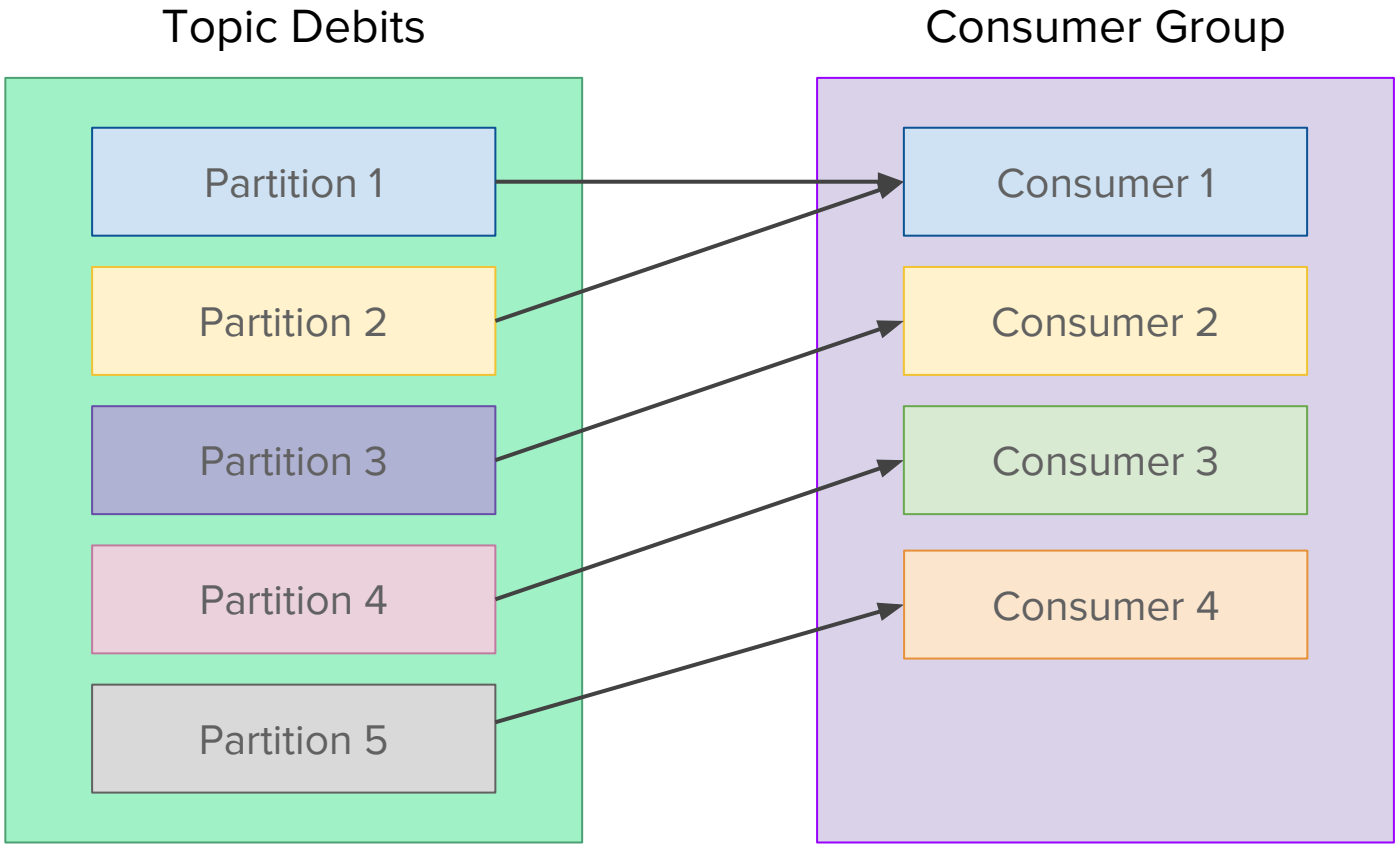
Consumer Group

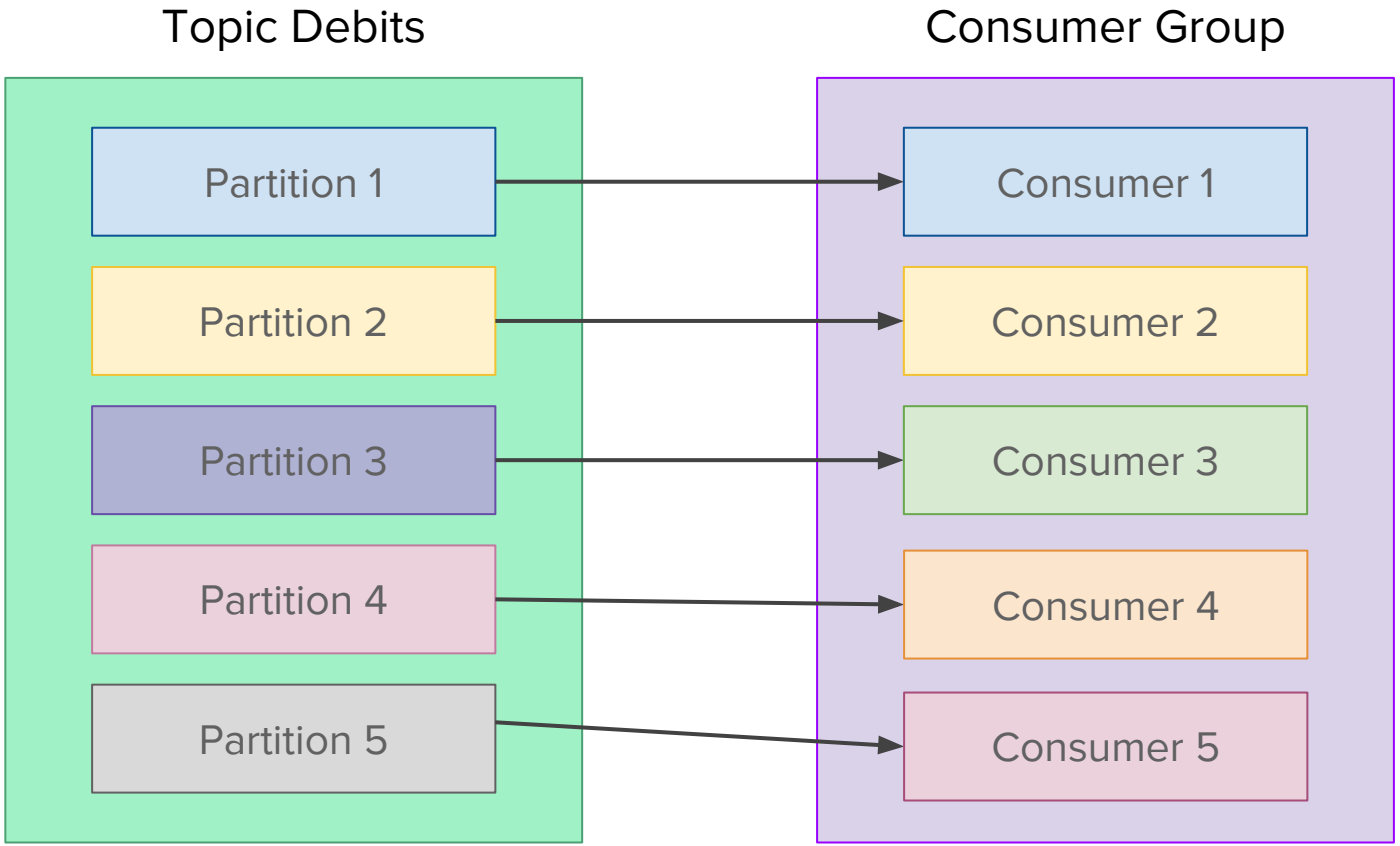
- Each consumer group is made up of 1 or more Consumers.
- Each message is delivered to one consumer within the Group.
- Each consumer will take ownership of a certain amount of partitions.




Topic Debits
Consumer Group
Partition 1
Partition 2
Partition 3
Partition 4
Partition 5
Consumer 1
Consumer 2


Topic Debits
Consumer Group
Partition 1
Partition 2
Partition 3
Partition 4
Partition 5
Consumer 1
Consumer 2
Consumer 3


Topic Debits
Consumer Group
Partition 1
Partition 2
Partition 3
Partition 4
Partition 5
Consumer 1
Consumer 2
Consumer 3
Consumer 4


Topic Debits
Consumer Group
Partition 1
Partition 2
Partition 3
Partition 4
Partition 5
Consumer 1
Consumer 2
Consumer 3
Consumer 4
Consumer 5

Quiz time

How many consumers in a Consumer group you should have for 5 partitions?

Ideally: # of Consumers at least less or equal than # Partitions

What will happen if you create more consumers than partitions?

The consumers will be idle for a period of time


OK! NOW WHAT?

Takeaway

- You can use it out of the box, you already know the Key Concepts ;-)
- Kafka is one of the Best Distributed Messaging Systems
- Scalable, Reliable, Fault tolerant
- Be mindful of you use cases
- Kafka API is really well documented
- There is multiple libraries for using Kafka clients:
  - Python, C++, Ruby, Golang, Java, Scala, Rust ....

Resources

- Kafka: a Distributed Messaging System for Log Processing
- Kafka Documentation
- Kafka the Definitive Guide

Sample & Slides

- github.com/laurauzcategui/wwcode-kafka

Q & A
Thanks

@laura_uzcategui