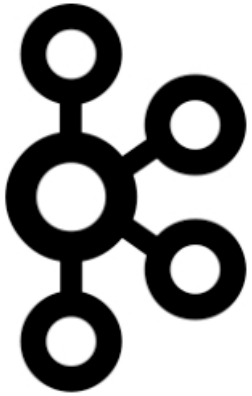
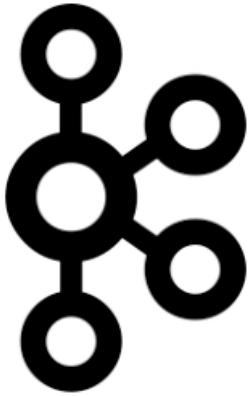
Microservices & Kafka



Kafka

Streaming | Messaging platform

Popular with Microservices developers



Overlapping of capabilities

Which one should be used for Microservices?

OR

Section objective

1. Kafka concepts
2. Kafka Vs. AMQP
3. Using Kafka in Microservices
4. Hands On : Kafka in action (JAVA)

Kafka 101

A quick introduction to Kafka

1. What is Kafka?
2. Kafka at LinkedIn
3. Core capabilities

Intent is to provide a high- level overview of Kafka!!!

Kafka

High performance open -source distributed event streaming platform

Messaging with topics

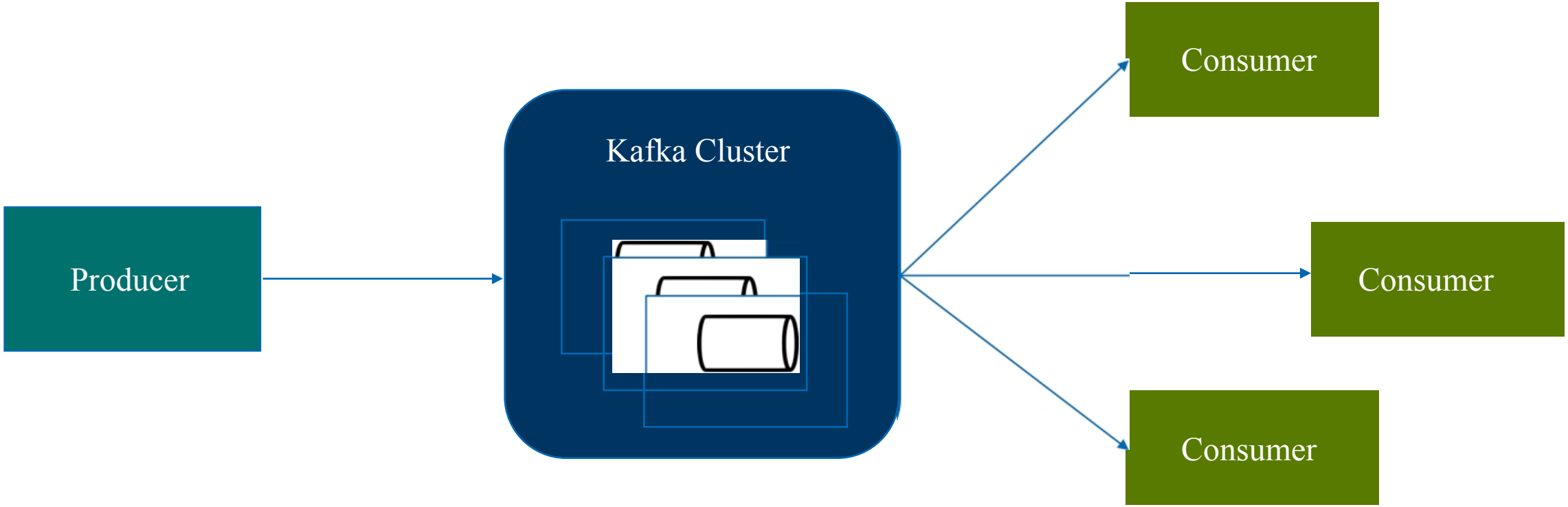
Multiple broker nodes in a cluster

2 Million writes/second

https://kafka.apache.org/

Kafka = Distributed messaging system

May consist of 1000's of machines across a network



May consist of 1000's of topics

Data persisted & replicated across thousands of partitions

Built for scale

Developed at Linked In & open sourced in 2011

- 7 Trillion messages / day

- 100 Clusters
- 4000 Brokers
- 100,000 Topics
- 7,000,000 Partitions

https://engineering.linkedin.com/blog/2019/apache - kafka- trillion - messages

Core capabilities

HIGH THROUGHPUT

Deliver messages at network limited throughput using a cluster of machines with latencies as low as 2ms.

SCALABLE

Scale production clusters up to a thousand brokers, trillions of messages per day, petabytes of data, hundreds of thousands of partitions. Elastically expand and contract storage and processing.

PERMANENT STORAGE

Store streams of data safely in a distributed, durable, fault-tolerant cluster.

HIGH AVAILABILITY

Stretch clusters efficiently over availability zones or connect separate clusters across geographic regions.

https://kafka.apache.org/

Quick Review

Kafka is a highly scalable & distributed messaging platform

Microservices use cases for Kafka

Kafka vs AMQP

Hands on experience with Kafka

Kafka Concepts

Ecosystem

BUILT-IN STREAM PROCESSING

Process streams of events with joins, aggregations, filters, transformations, and more, using event-time and exactly-once processing.

CONNECT TO ALMOST ANYTHING

Kafka's out-of-the-box Connect interface integrates with hundreds of event sources and event sinks including Postgres, JMS, Elasticsearch, AWS S3, and more.

CLIENT LIBRARIES

Read, write, and process streams of events in a vast array of programming languages.

LARGE ECOSYSTEM OPEN SOURCE TOOLS

Large ecosystem of open source tools: Leverage a vast array of community-driven tooling.

https://kafka.apache.org/

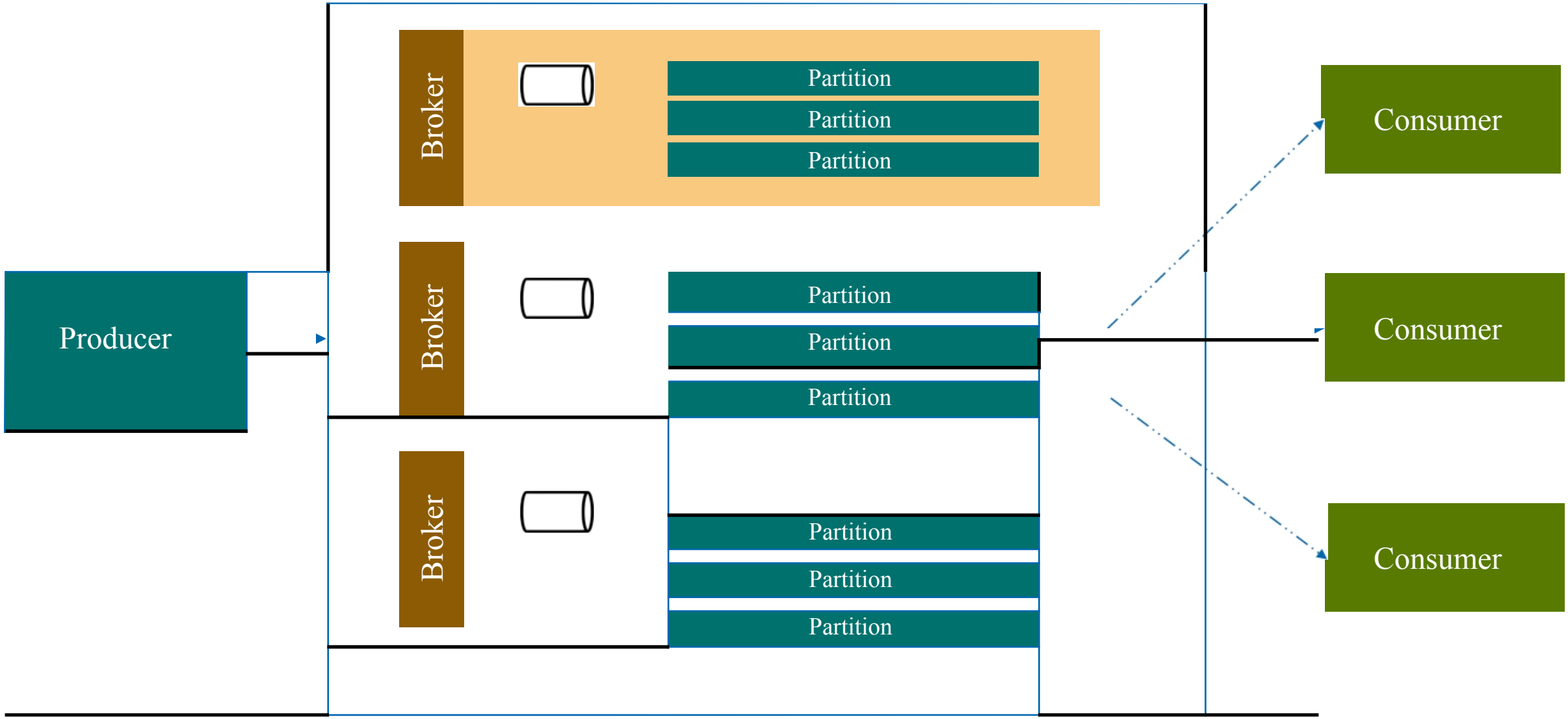
Kafka Concepts

Common terms used in Kafka

1. Broker nodes, Leader
2. Topics, Partitions, Offset
3. Retention period, Group ID

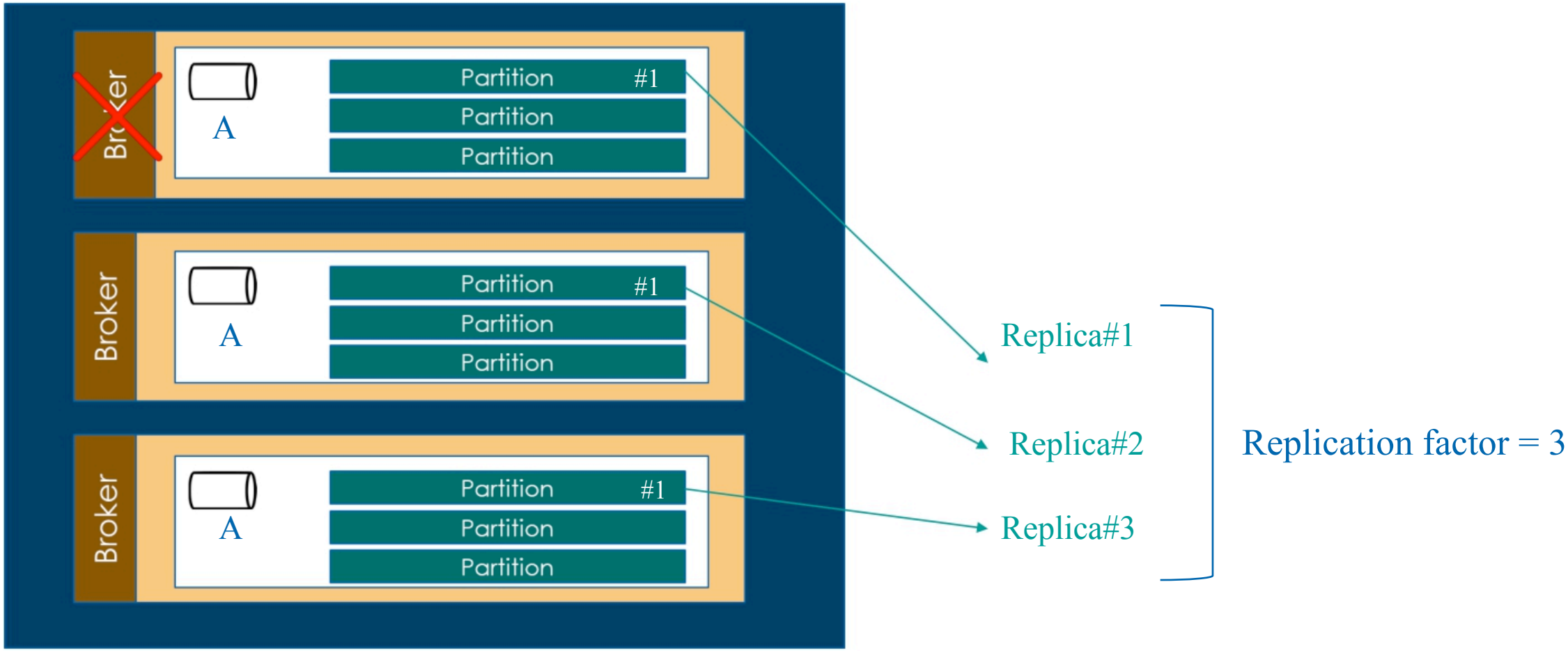
Intent is to provide a high- level overview of essential concepts !!!

Kafka Cluster



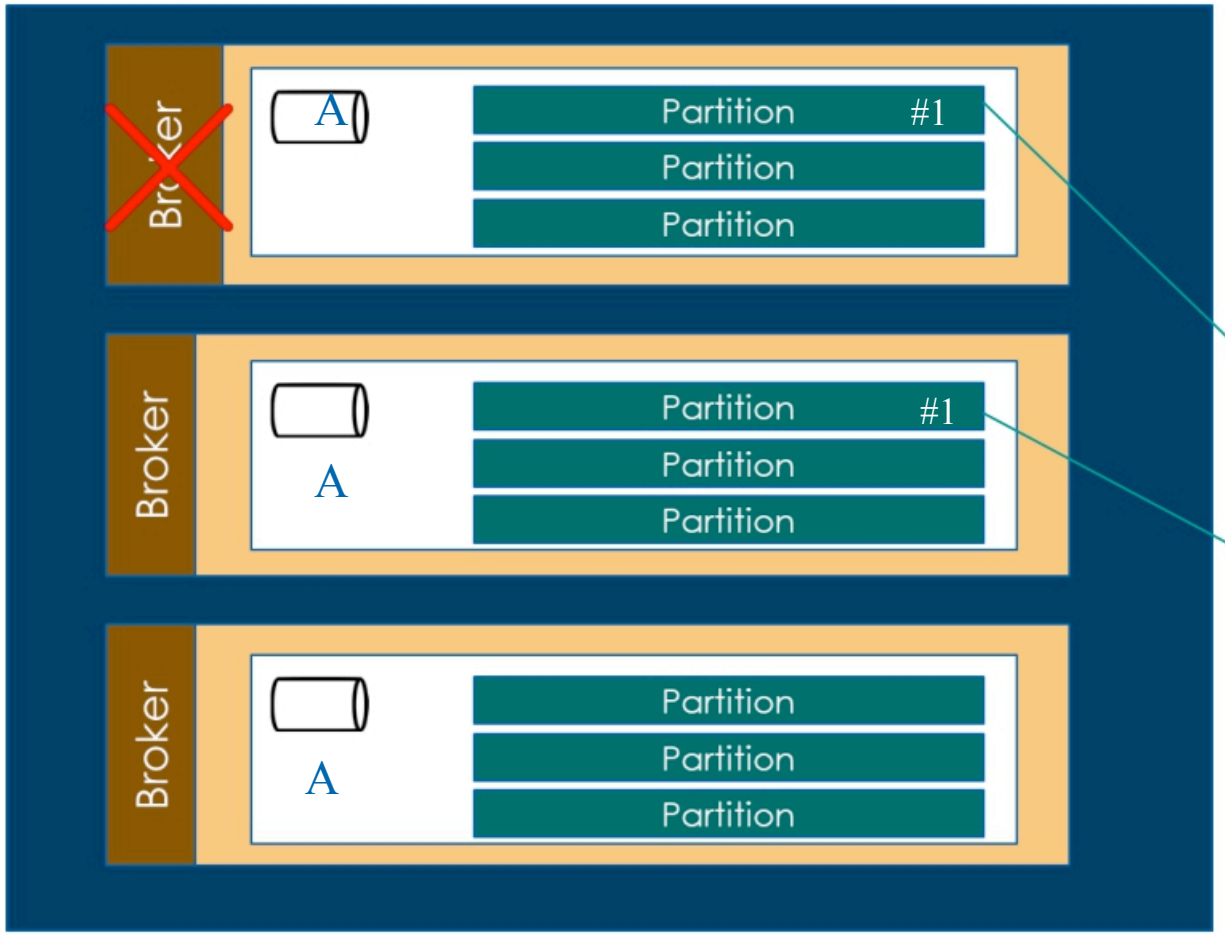
Event Message data

Replicated across multiple Brokers in the partitions



Event Message data

Replicated across multiple Brokers in the partitions



- Broker failure does not impact the cluster

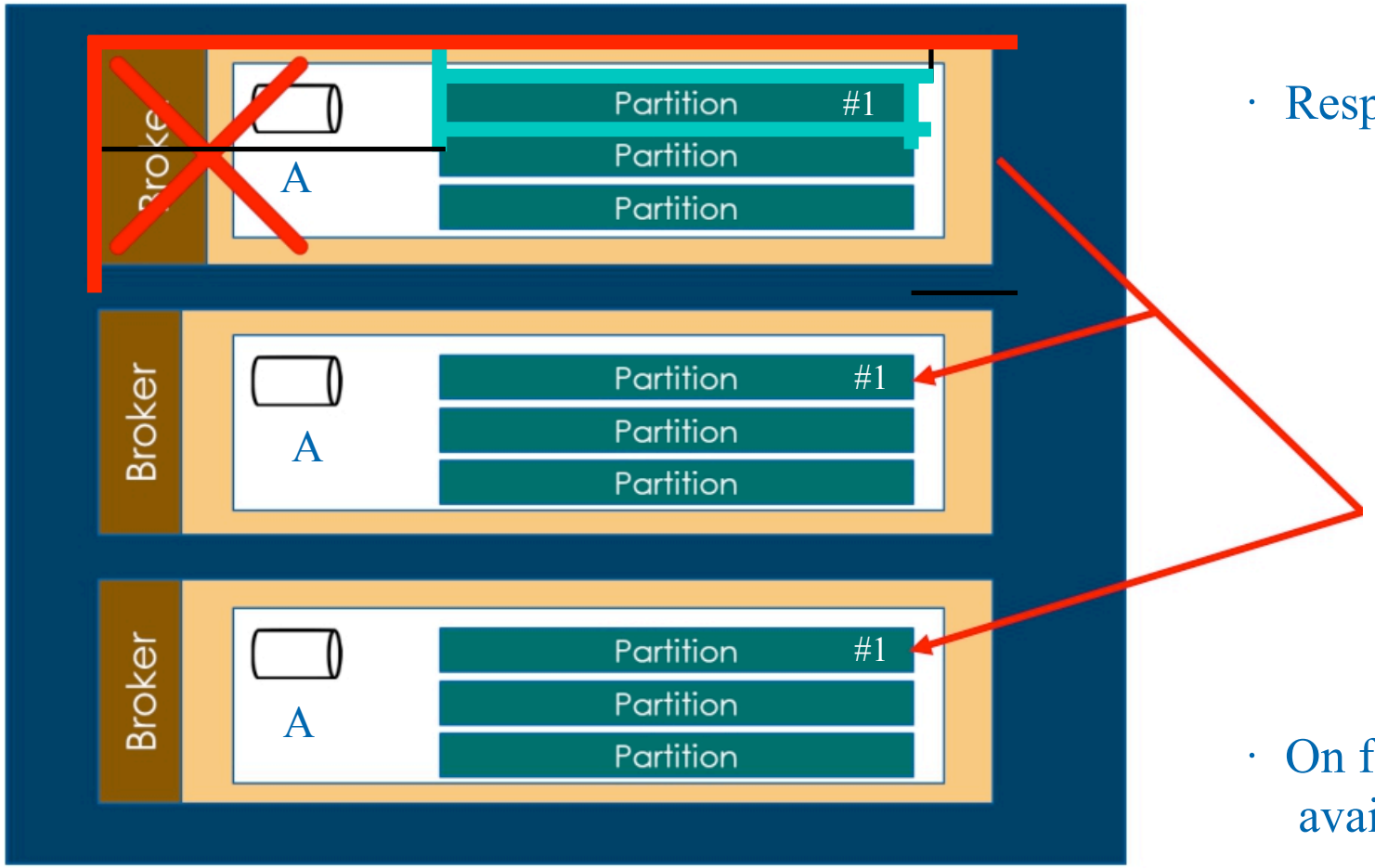
Replica#1

Replica#2

Replication factor = 2

Leader

One node act as a Leader for a partition



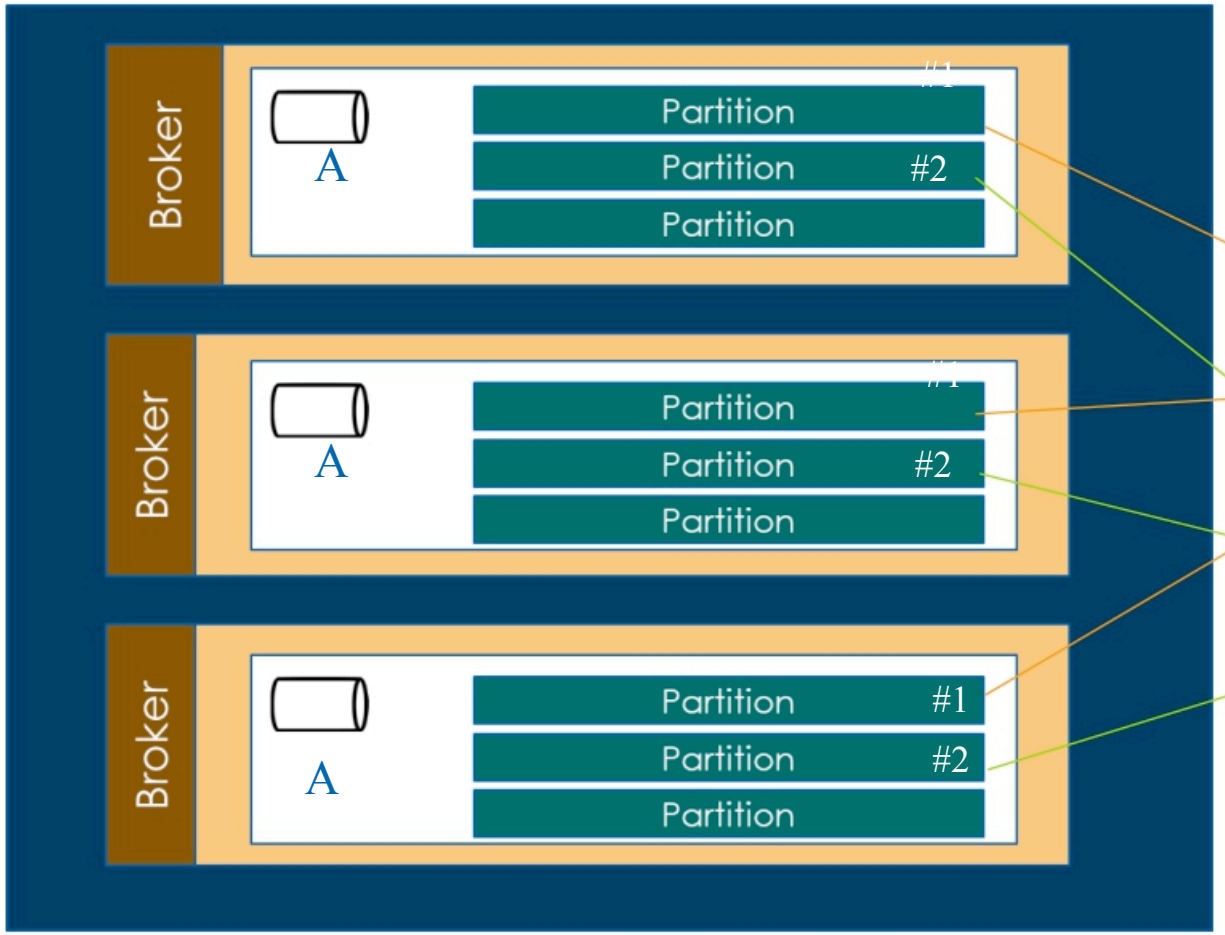
- Responsible for all Reads & Writes

Leader replicates the data

- On failure of Leader one of the available broker becomes the leader

Partitions

Message Data is partitioned on message key



- Producer may provides the key with data

key = customer- id - 123

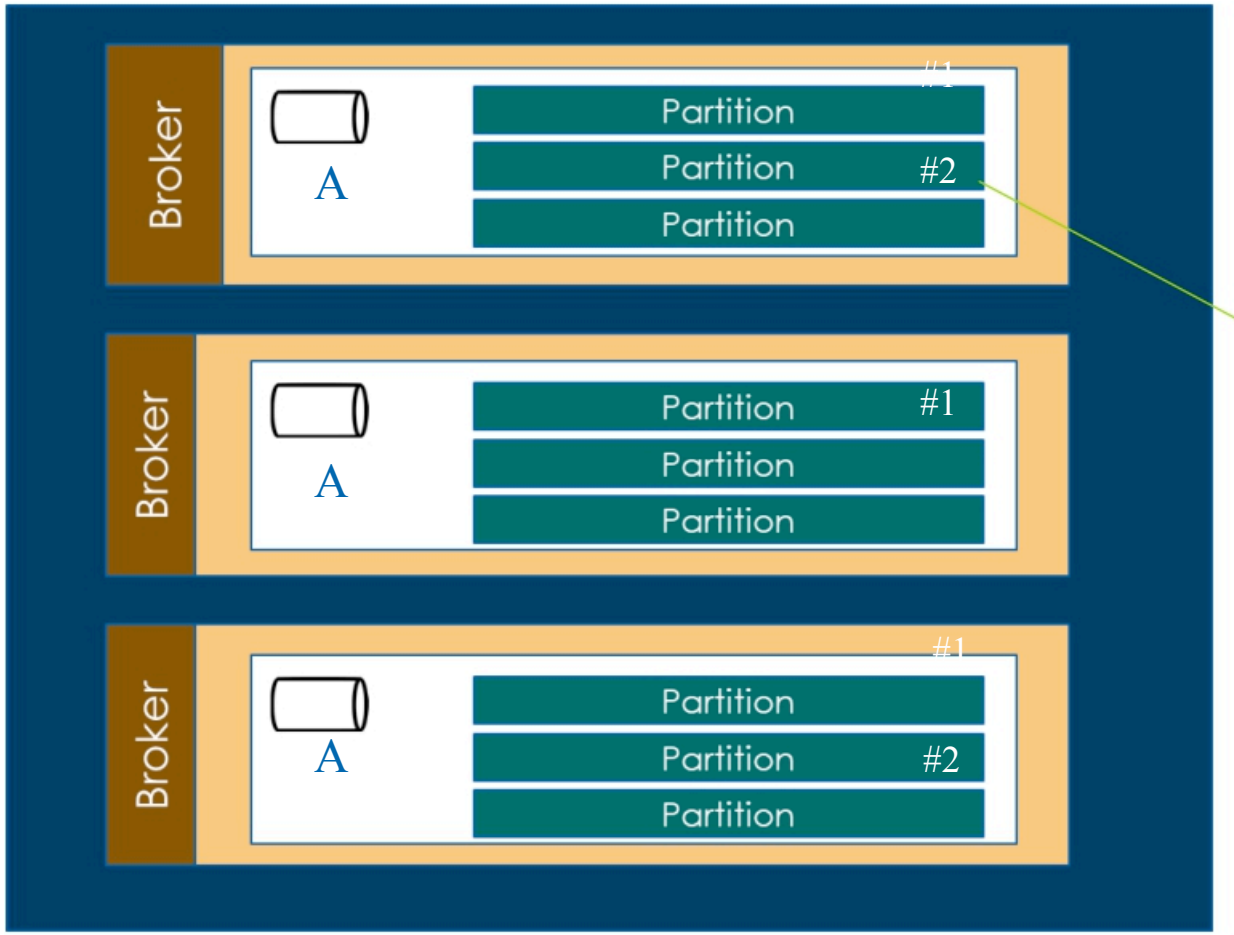
key = customer- id - 789

- No Key specified

Partition assignment is round robin

Message Ordering

Messages are ordered ONLY within a partition



- NO order guarantee across partitions

Consumer

Message order maintained

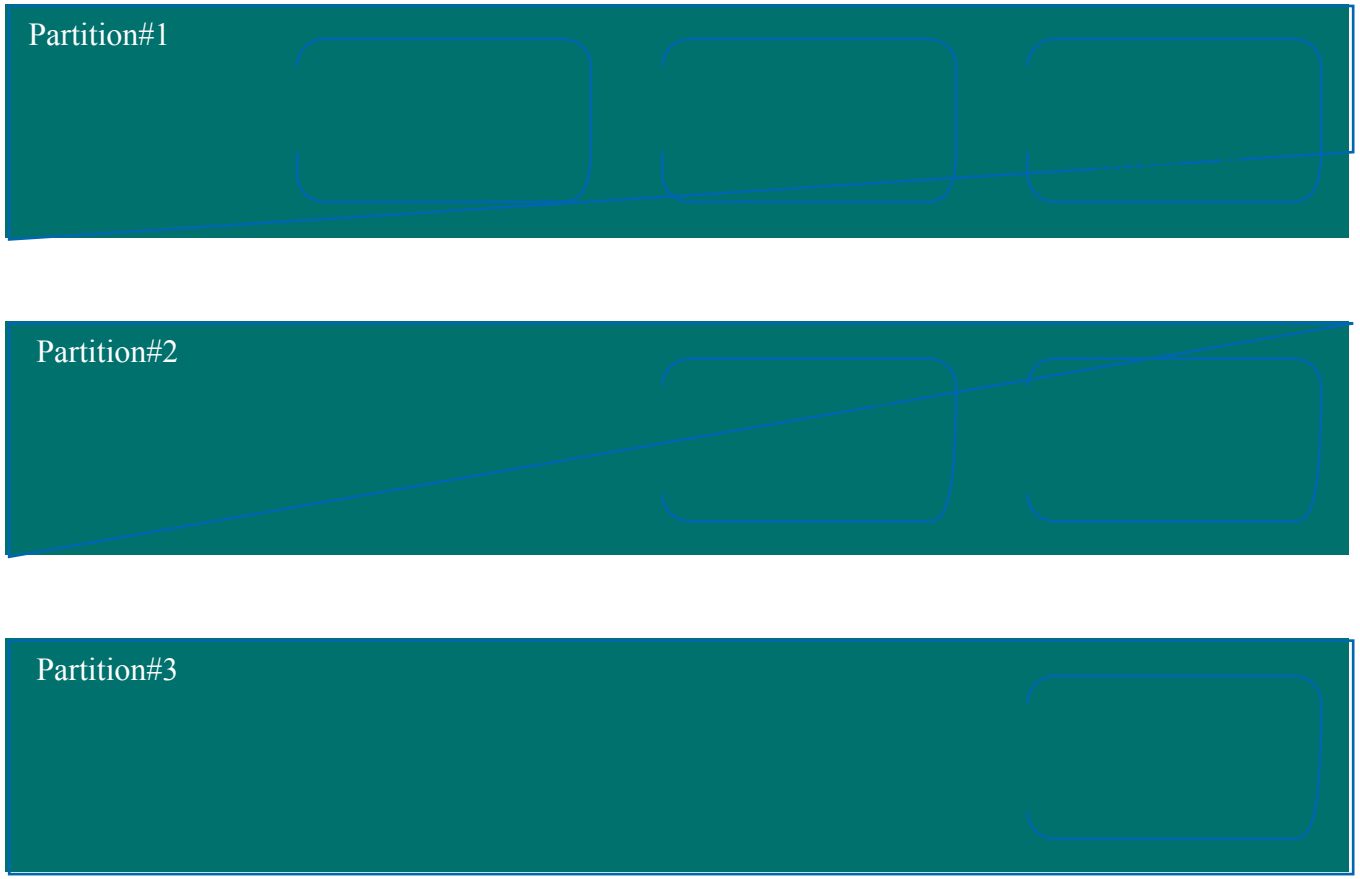
Producer |Publishing

Each message in a partition is assigned an offset

Producer

```
produce {
        key =

        data = …
}
```



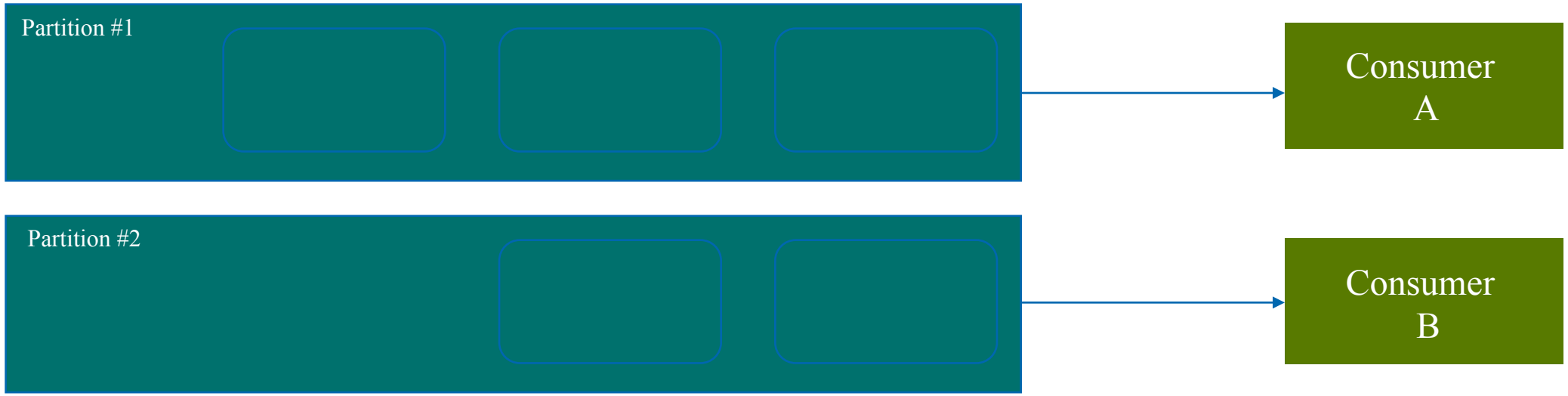
Consumer |Subscription

Consumer polls for the messages; fixed interval

- Acknowledges the message after read
- Unlike AMQP - Message is **NOT** removed
    - Kafka updates read offset for that consumer

Consumer |Subscription

Maximum of 1 active consumer per partition



Consumer	Partition	Offset
A	1	
B	2	

Kafka maintains the current position of the consumer to avoid duplication

Consumer |Subscription

Reads are NON -destructive

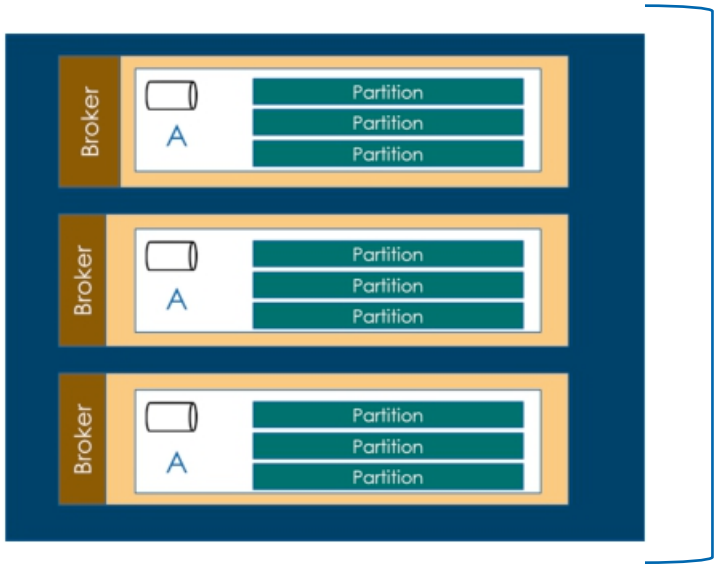
- Consumer can Reset the offset
- Consumer can replay the log/messages from any offset
- Messages expire after the retention period

Consumer may manage the offset on it's end instead of relying on Kafka

Consumer |Subscription

Multiple consumer instances can listen to a common topic

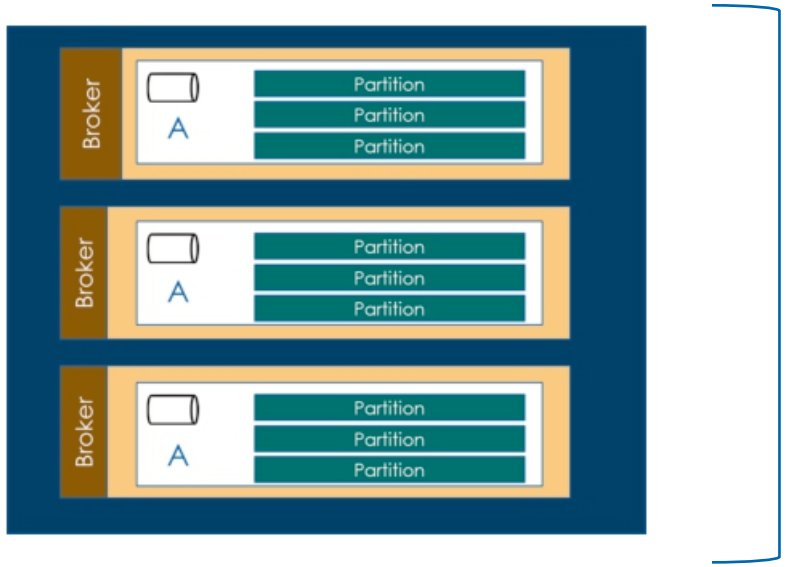
- Each message to be received by ONLY one consumer in the group
  - Such consumers are grouped by way of group ID



Common group ID assigned to each consumer

Consumer |Subscription

Max number of consumer in a group = Number of partitions



- Max of 3 consumers in a Group

Quick Review

Reads are nondestructive

Topics have partitions; message key determines the partition

Kafka manages the offset; consumer can reset offset

Consumers can read as a group by way of group ID

Kafka cluster setup

Setup a Kafka cluster on the cloud

1. Setup a Kafka cluster on CloudKrafka (free)
2. Walkthrough of console

Please follow along to create the Kafka cluster !!!

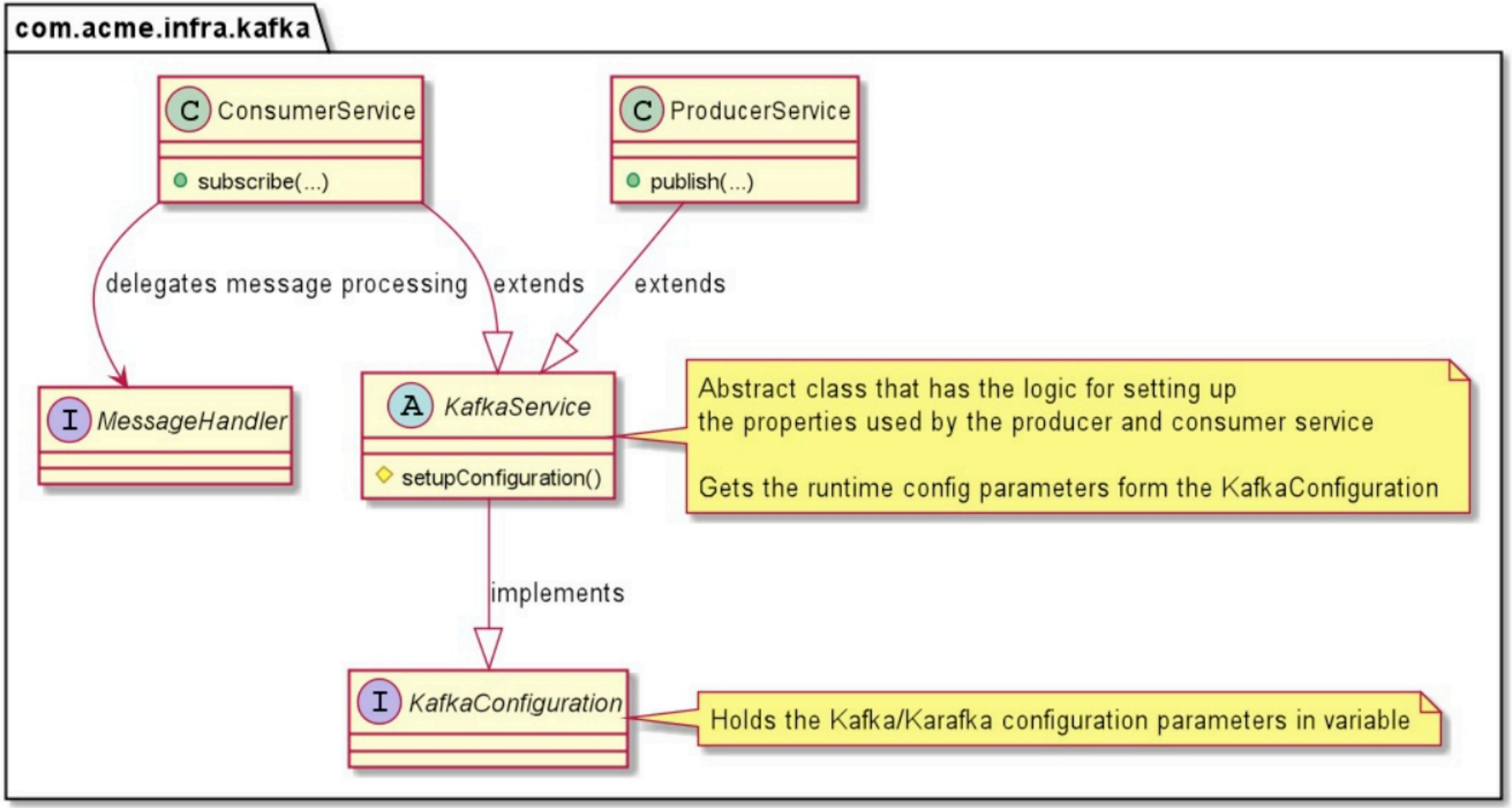
Kafka producer in action

Publish messages to Kafka topic

1. Class diagram walkthrough of utility classes
2. Quick look at the producer service class code
3. Kafka producer in action

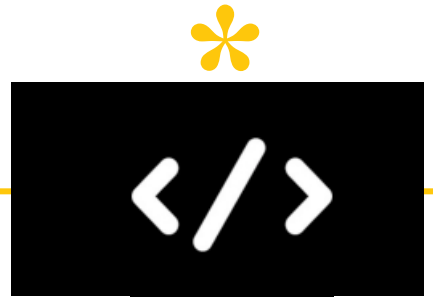
Kafka Utility Classes

uml/ kafka/ kafka.class.puml



Kafka consumer group in action

Receive messages in a consumer group



1. Kafka rebalancing protocol
2. Quick look at the code
3. Consumer group in action

Kafka Rebalancing Protocol

ONE consumer per partition

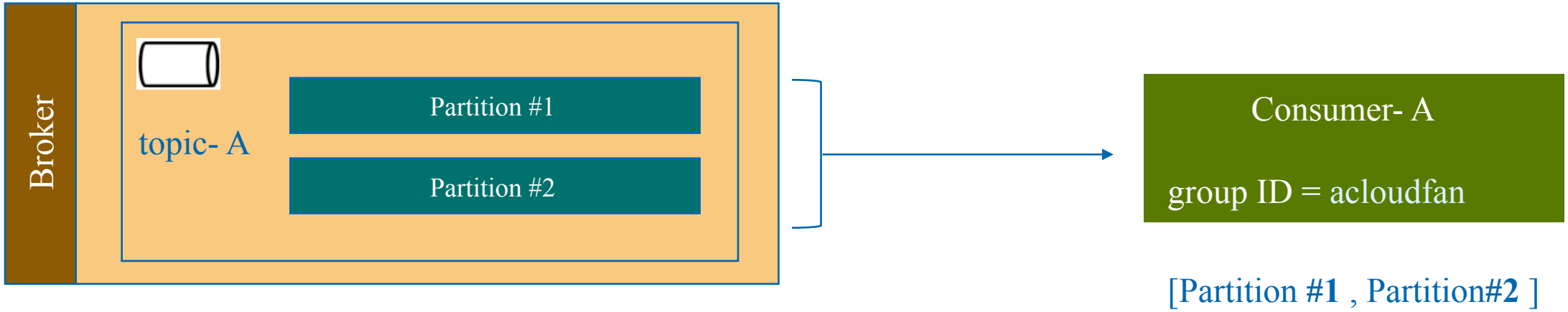
Kafka ensures that:

Each consumer in a groups attach to 0 or more partitions

Consumer group automatically re -assign partitions

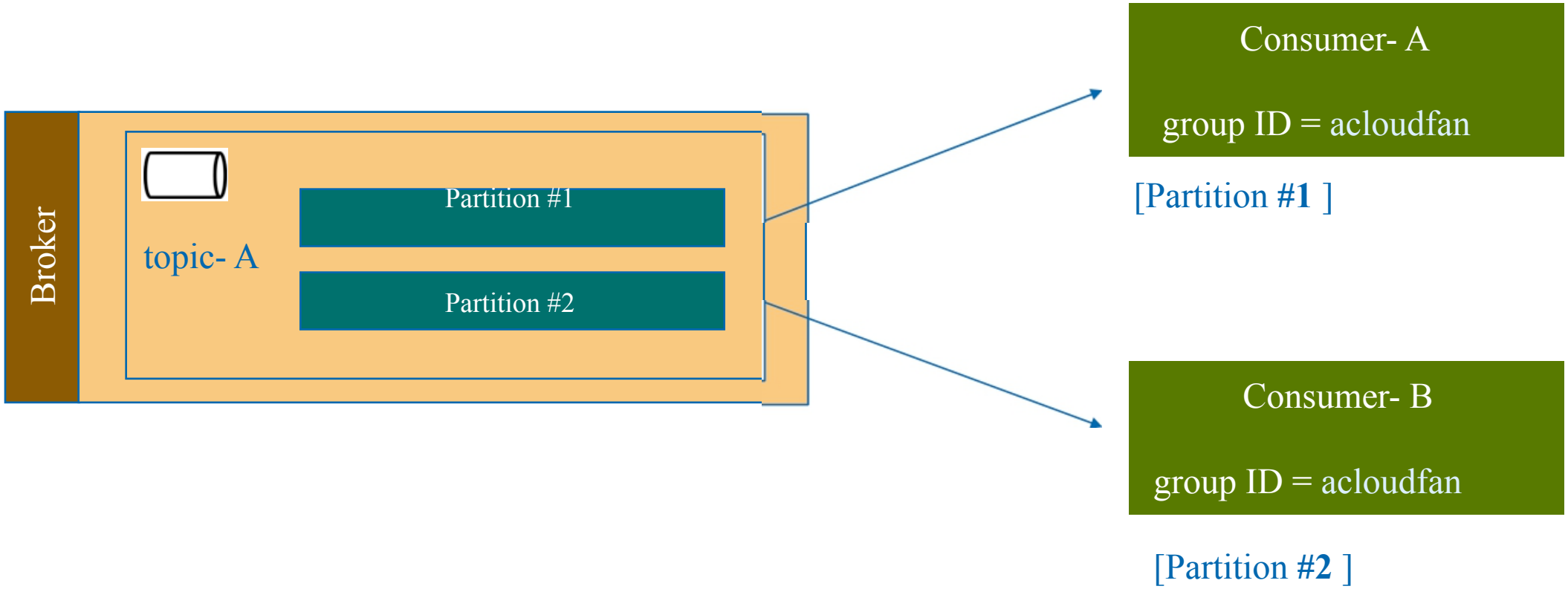
Consumer Rebalancing

Kafka consumers in a group are automatically re -balanced



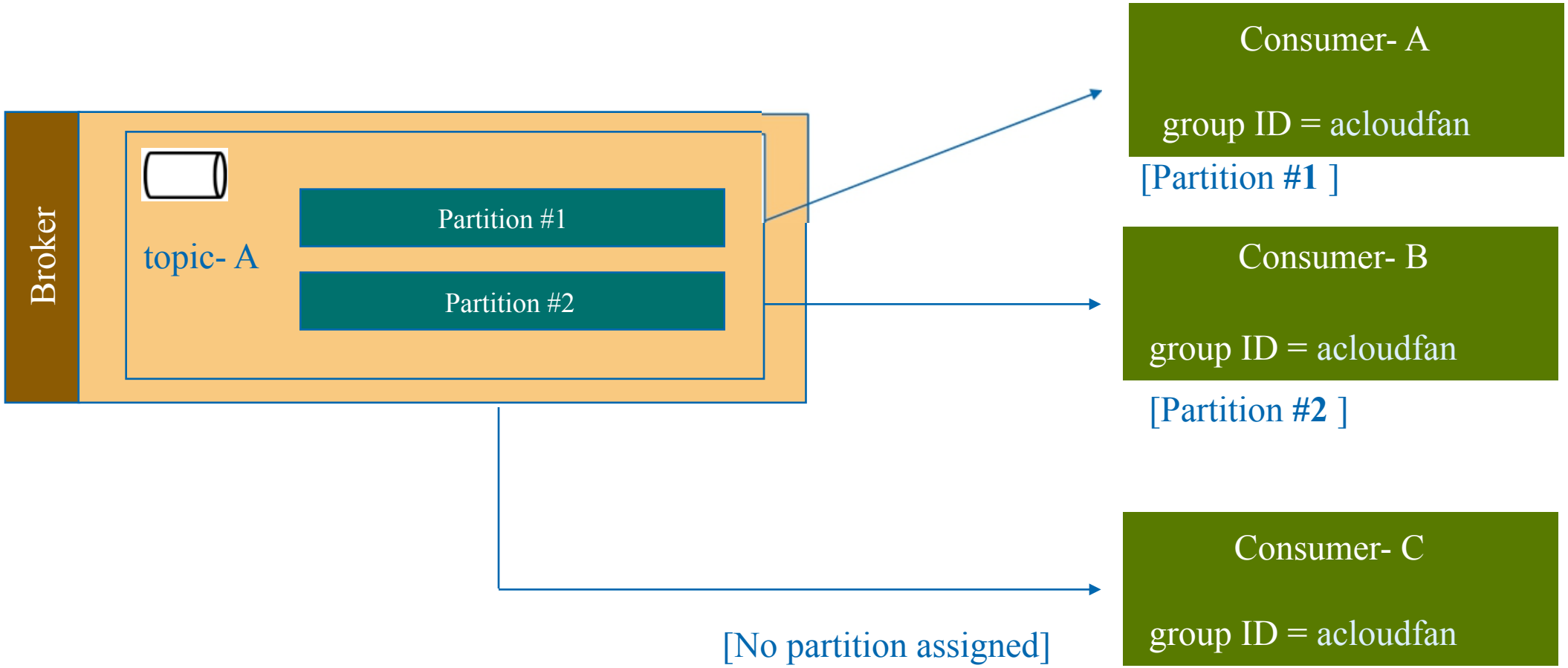
Consumer Rebalancing

Kafka consumers are reconfigured automatically



Consumer Rebalancing

Consumer placed in wait state if number of consumers exceed number of partitions

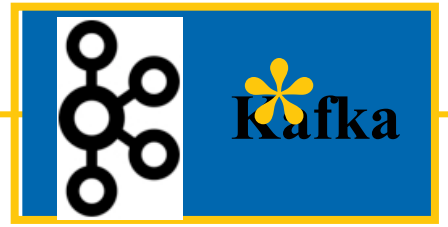


Testing steps : Kafka rebalancing

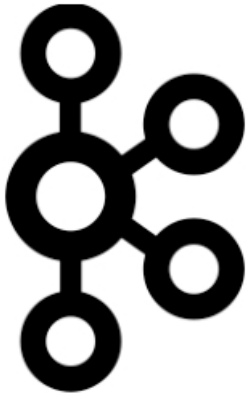
1. Create a new topic with 2 partitions
2. Setup 2 consumer/subscriber instances
3. Publish messages with different *keys*

Kafka versus AMQP

Differences between the messaging platforms



1 A quick comparison



Apache ACTIVEMQ

RabbitMQ

What would you use for Microservices?

Kafka versus AMQP

Kafka always persists the message

- Message automatically expires after the set duration

- Messages are not removed on READ so can be replayed

RabbitMQ Removes message from Queue on Read

Kafka versus AMQP

Kafka uses a custom protocol

- Binary protocol over TCP
- Protocol used by Kafka is NOT a standard

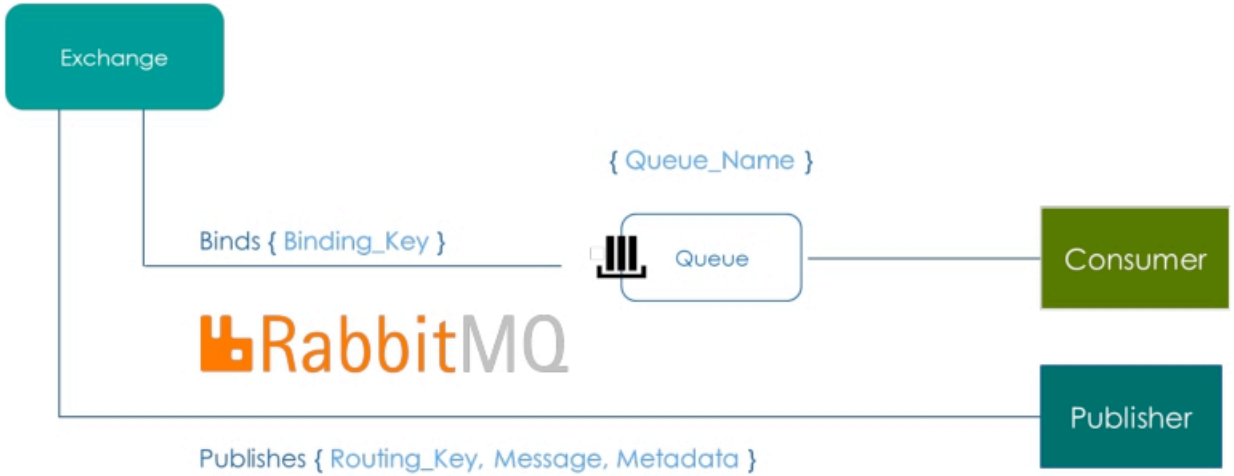
Implements the AMQP (protocol)

Kafka versus AMQP

Kafka does not support routing

- Dynamic routing is possible but not out of the box

- AMQP has a very flexible routing mechanism



Kafka versus AMQP

Kafka has no concept of Queue

- Supports ONLY publish - subscribe messaging pattern

- AMQP support point- to - point and publish- subscribe patterns

Kafka versus AMQP

Kafka consumers ALWAYS pull messages from broker (polling)

- Consumer poll for the specific duration & receive batches

- AMQP/RabbitMQ supports push and pull

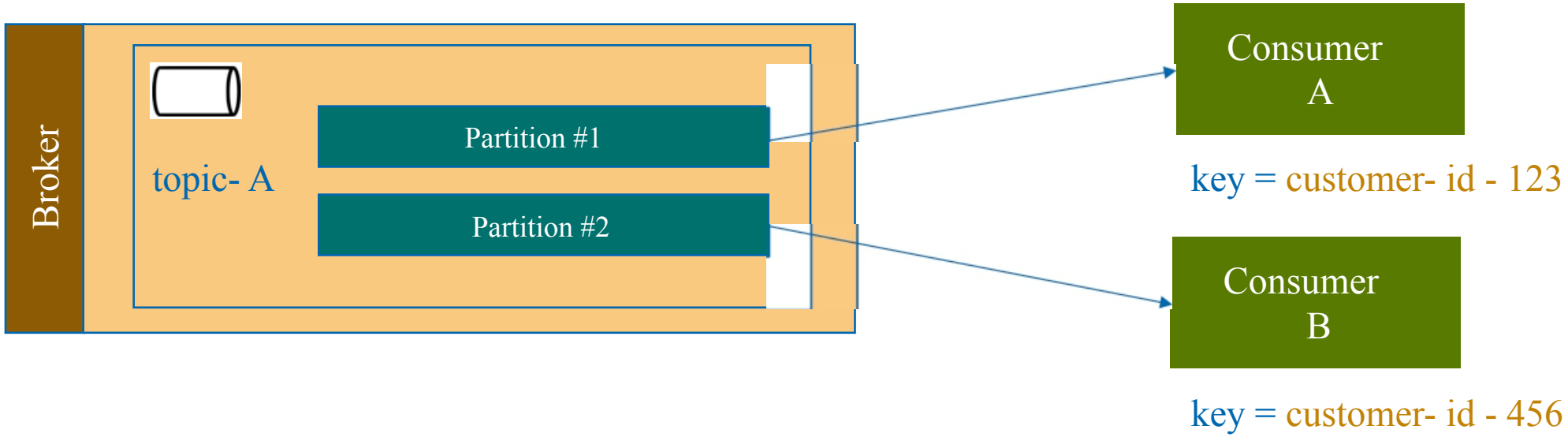
Kafka versus AMQP

Kafka does not have the concept of:

- Exchanges, Queues, Bindings
- Message priority

Kafka versus AMQP

Kafka by default guarantees message ordering per partition



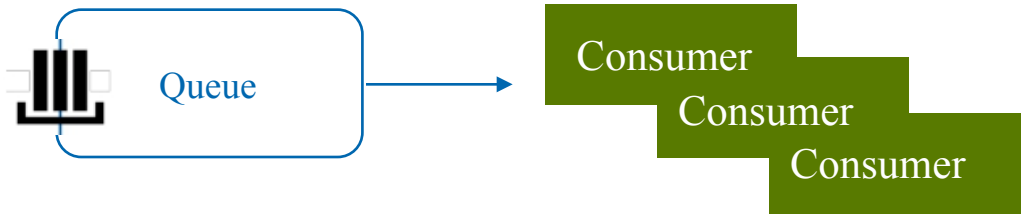
- Consumer receives messages in order they were published

Kafka versus AMQP

Kafka by default guarantees message ordering per partition

RabbitMQ

- Message in a queue may be read by multiple consumers



- Processing order is not guaranteed !!!

Quick Review

1. Kafka message reads are nondestructive

2. Kafka does not support Queue, Exchange, Routing, Priorities ….

3. Kafka uses a custom binary TCP protocol

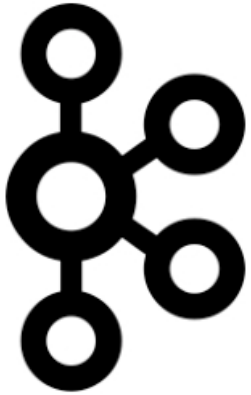
4. Kafka supports ONLY pub - sub and pull based message receive

5. Kafka guarantees message order within a partition

Kafka for Microservices

Messaging for microservices

1. Common use cases for Kafka
2. Scenario based questions (& answers)



Apache ACTIVEMQ

RabbitMQ

What would you use for Microservices?

Depends on the use cases for your microservice

What would you use for Microservices?

Kafka use cases

Large scale messaging applications

- Strong message durability
- Unlimited scaling by way of topic | partitions

https://kafka.apache.org/uses

Kafka use cases

Web site tracking a.k.a. click stream analysis

- Site activity requires high ingestion rate; one topic per activity
- This was the original use case for Kafka

https://kafka.apache.org/uses

Kafka use cases

Log aggregation & Stream processing

- Collection of logs from servers/application across 1000s of servers
- Data received as event streams that may be analyzed in Realtime

https://kafka.apache.org/uses

Kafka use cases

Event sourcing | Kafka is a natural Event Store

- Time ordered sequence of records
- Event data is retained in Kafka and may be replayed

https://kafka.apache.org/uses

Kafka use cases

Commit Log

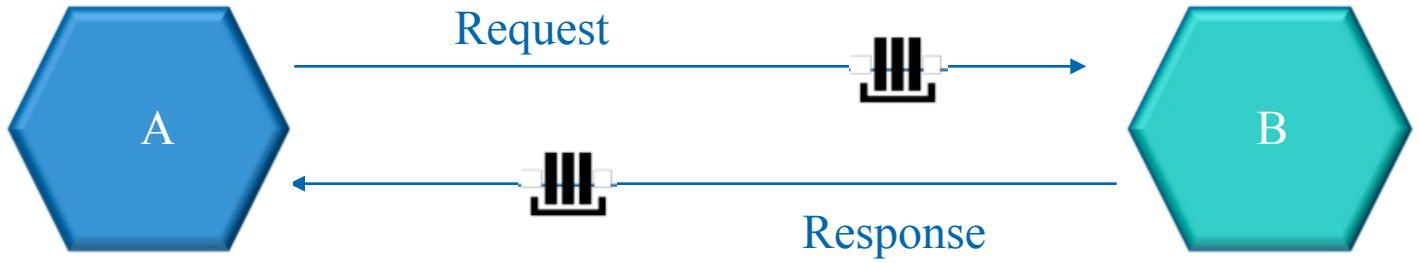
- Provides commit - log for a distributed system
- Supports replication of data between nodes; re - syncs failed nodes

https://kafka.apache.org/uses

Question & Answers time !!

RabbitMQ

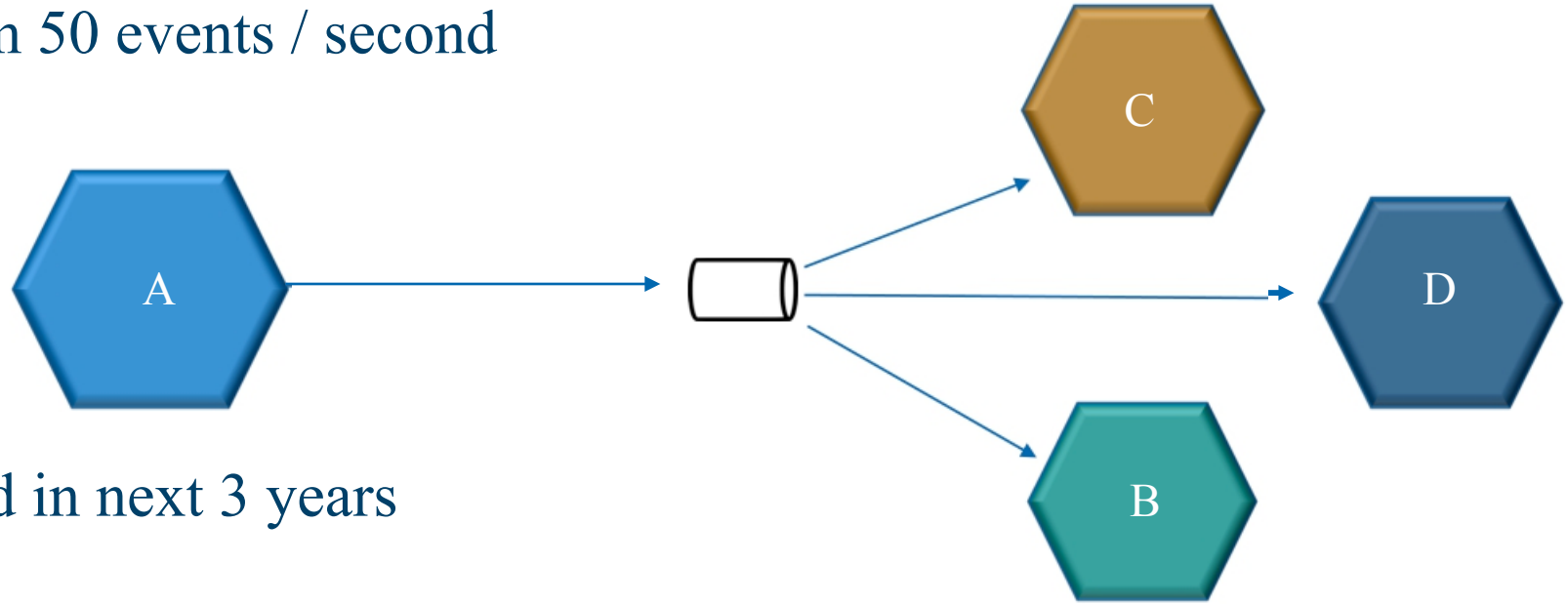
Which one would you prefer & why?



What would prefer & why?

RabbitMQ

Maximum 50 events / second

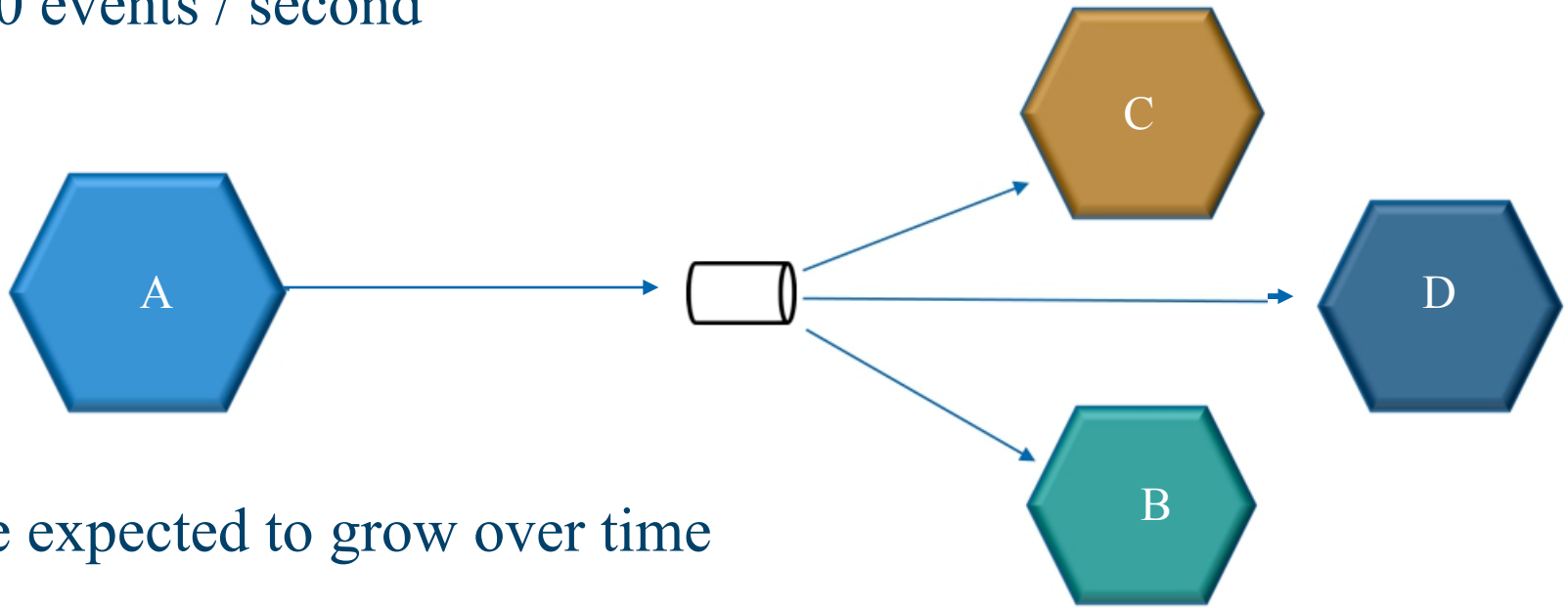


NO change expected in next 3 years

What would prefer & why?

RabbitMQ

50,000 events / second

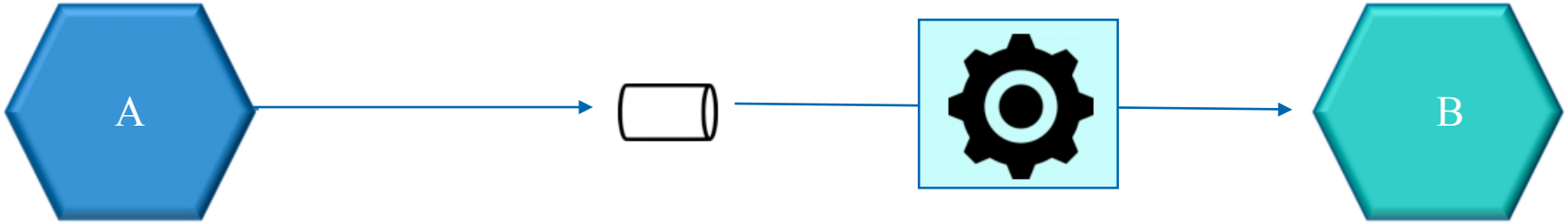


Volume expected to grow over time

What would prefer & why?

RabbitMQ

1000 events / second



Real time events analysis

E.g., anomalous credit card charges

What would prefer & why?

RabbitMQ

- Provides a standard based messaging interface
- Consumers of the interface prefer simplicity

What would prefer & why?

Assume: Messaging requirements may be fulfilled by both

As a developer I would like to have full control on routing of event data