Kafka

Prudhvi Akella
@Software Engineer – Big data Analytics

© 2019 Prudhvi Akella

Road Map

- What is Kafka?
- How Kafka works?
- Why Kafka?

© 2019 Prudhvi Akella

Lets Start

● What is Kafka?

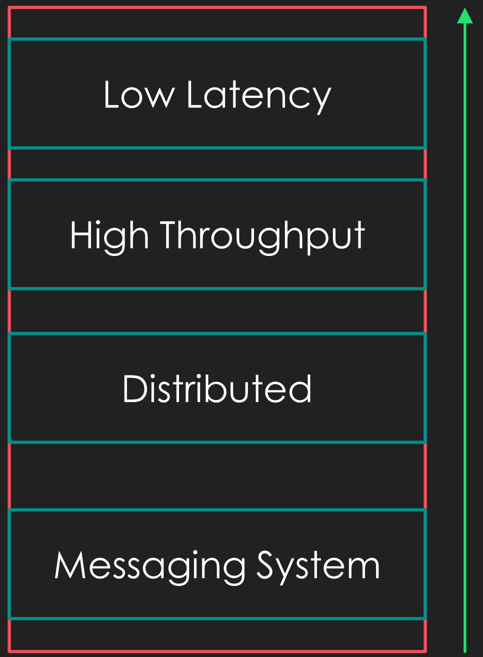
●

●

© 2019 Prudhvi Akella

Definition

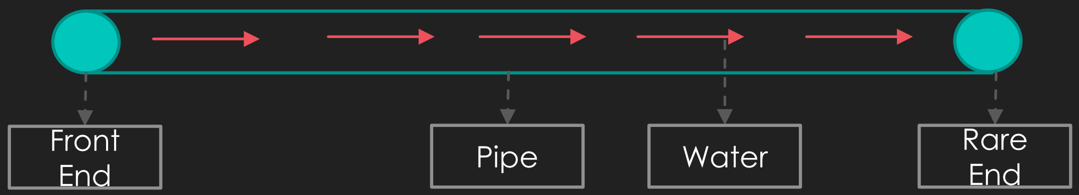
Kafka is a High Throughput Distributed Messaging System used to build Low Latency System.

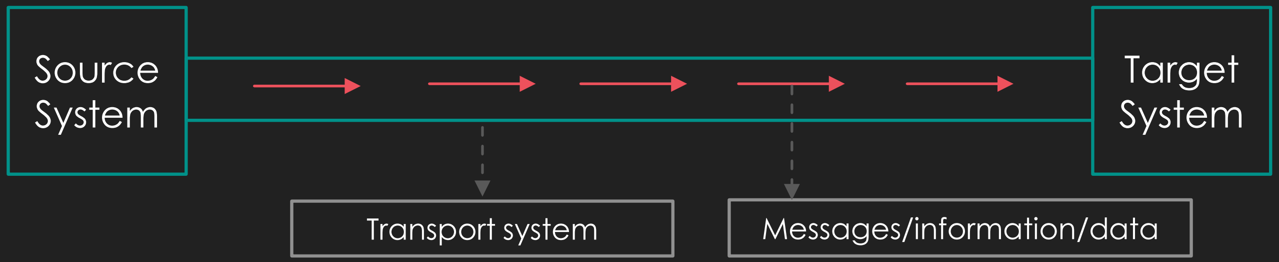


Lets talks about each key word in detail from down to top approach

© 2019 Prudhvi Akella

Messaging System

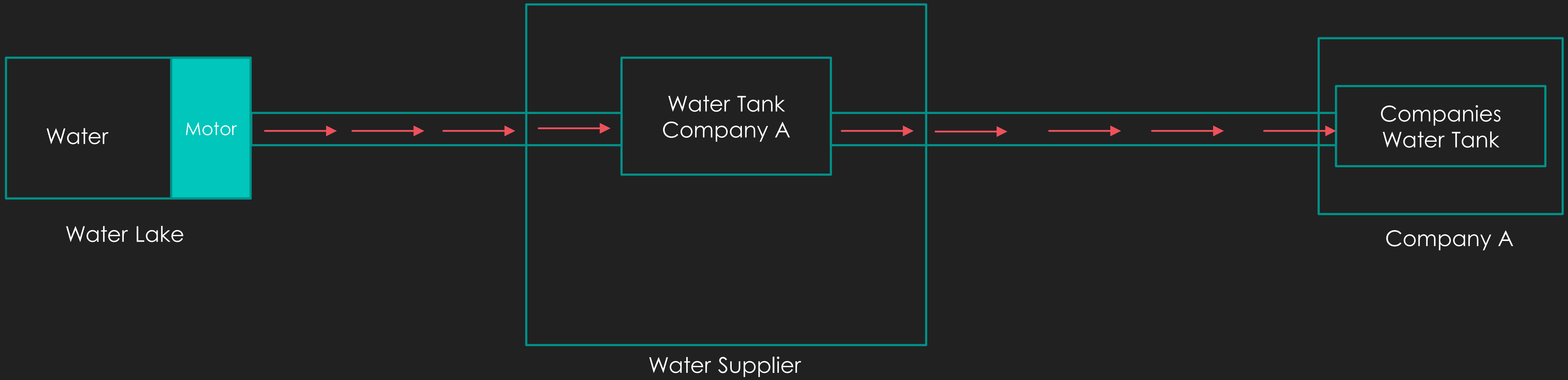




In the way how pipe is used to transfer the data b/w front end and rare end in the same way in computer science terms a messaging system acts like a transport system/channel which is used to transfer the data b/w source system and target system

© 2019 Prudhvi Akella

Lets Talk about distributed with an Example



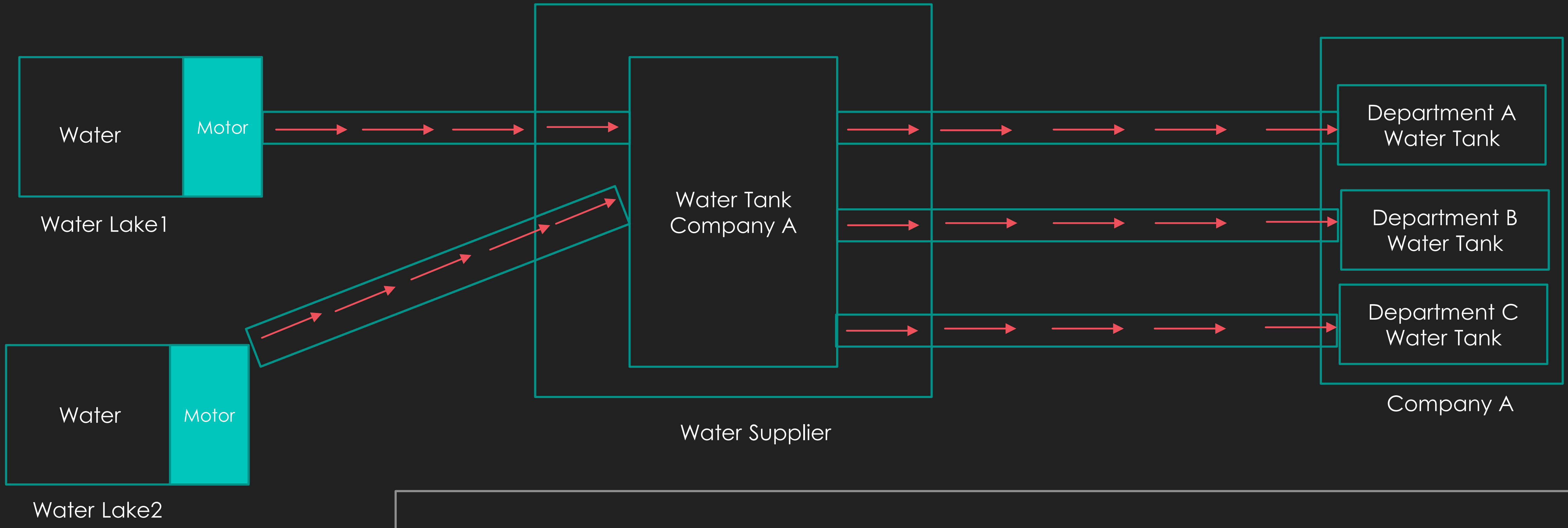
There is a company A which needs water for there internal processing so they approach water supplier then water supplier construct a separate water tank for company in his eco system and install a motor with a pipelines at the lake which connect to water tank and supplies water to the company. All good.

Now challenges

1) What if water in the lake gets exhausted
2) Company has three departments all need water at same time but as there is only one tank in the company departments are distributing the water in schedule basis that means 5-10 1-5 5-10 due to that production is effecting. Now how to solve this problem

© 2019 Prudhvi Akella

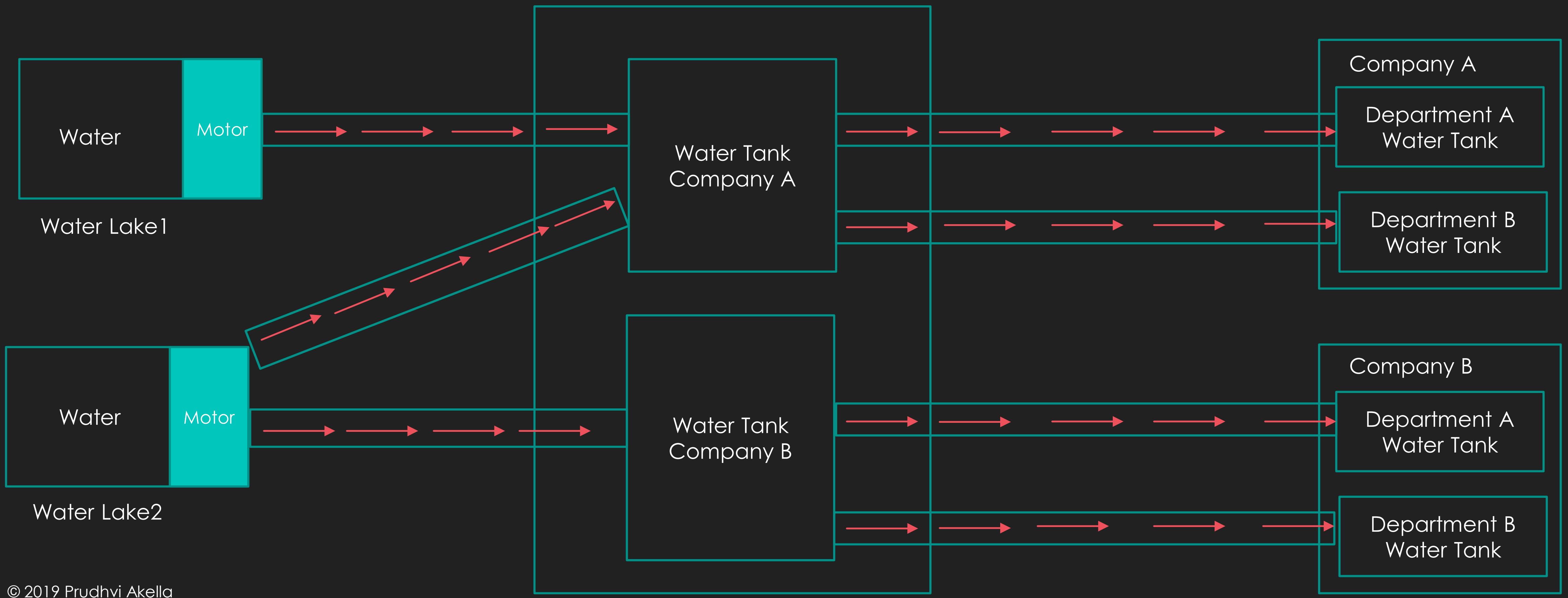
Lets Talk about distributed with an Example



What if new company approach's supplier? What will be do ?

© 2019 Prudhvi Akella

Lets Talk about distributed with an Example



© 2019 Prudhvi Akella

Lets talk about High Throughput

Everything looks good till now correct water is flowing from lake to company through a tank.
But Challenge is company is not receiving water at the rate it is expected lets it is suppose to receive 100 liters per second but it is at the rate of 60 liters per second.
What to do and what are the possible ways to increase the speed?

© 2019 Prudhvi Akella

Lets talk about High Throughput

Everything looks good till now correct water is flowing from lake to company through a tank.
But Challenge is company is not receiving water at the rate it is expected lets it is suppose to receive 100 liters per second but it is at the rate of 60 liters per second.
What to do and what are the possible ways to increase the speed?

- Increase the pipe size from lake to water supply tank and water supply tank to department tank
- Add a motor b/w the water supply tank and company tank
- Increase the motor capacity b/w lake and water supply tank

Throughput is nothing but at what rate the data is getting transmitted between source to destination
Lets say 1 MB per second
2 MB per second
When you see high in prefix it means we are talking about some big data system. because big data system needs high throughput

© 2019 Prudhvi Akella

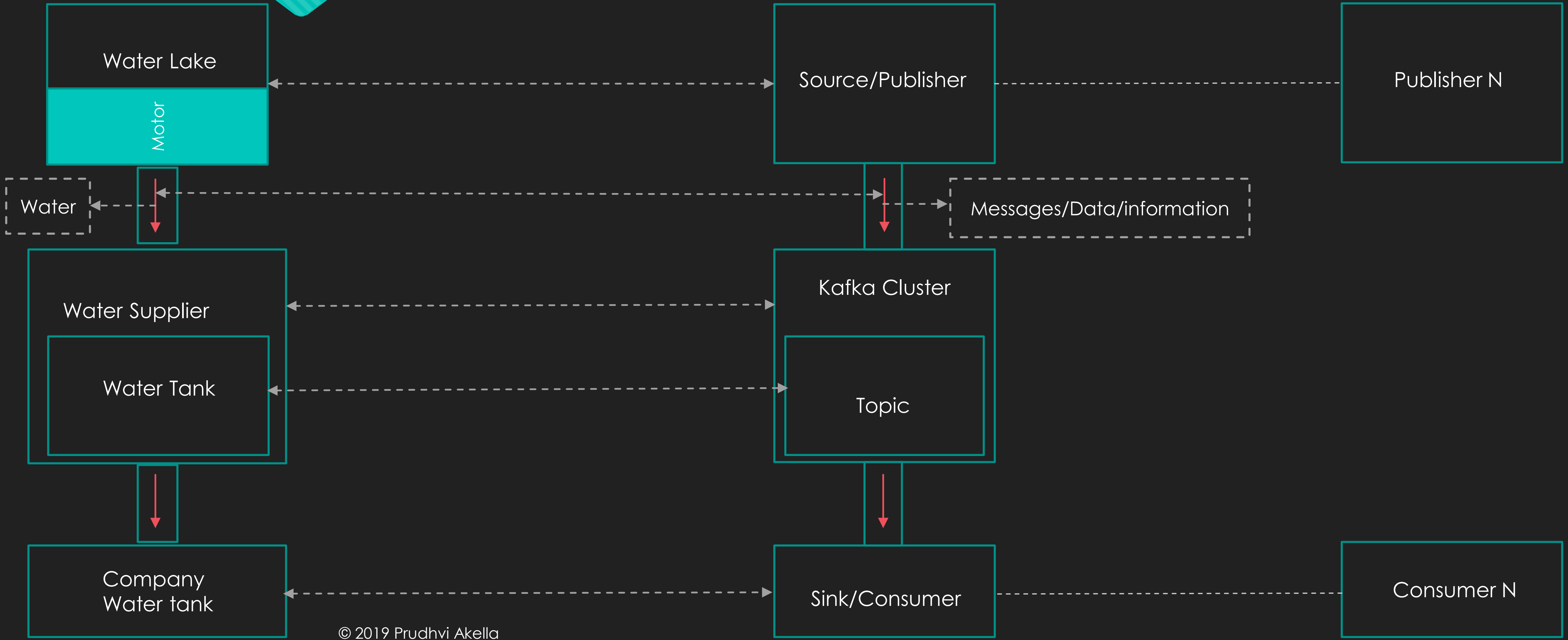
Lets talk about Low latency

Latency is nothing but how much time water takes to transfer from lake to the company tank.

The time taken by a data to transfer from source system to target system it called latency. In big data system the latency has to be low if it is less than 0.10 ms then it is considered to be real time application.

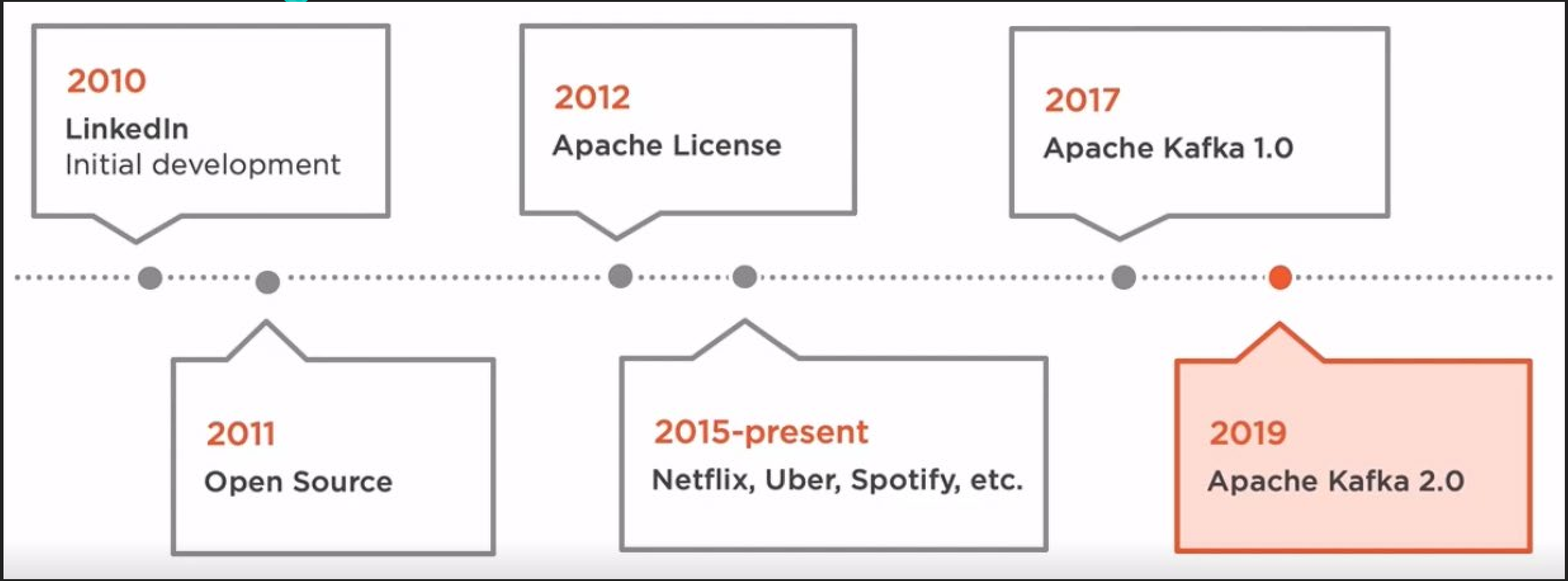
© 2019 Prudhvi Akella

High Level view of Kafka



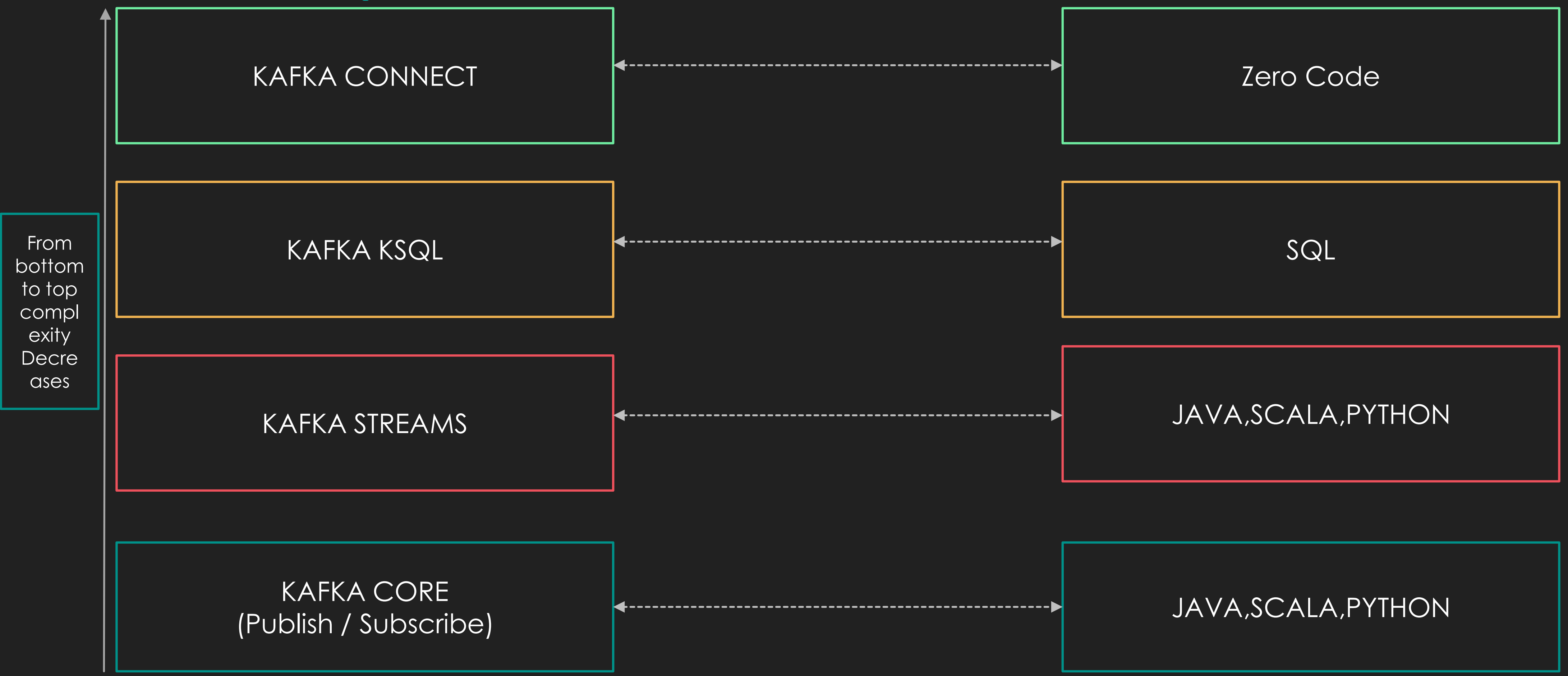
© 2019 Prudhvi Akella

Kafka History



© 2019 Prudhvi Akella

Who all can work with Kafka?



© 2019 Prudhvi Akella

Any doubts till now?

© 2019 Prudhvi Akella

Lets Start

- What is Kafka?
- How Kafka works?

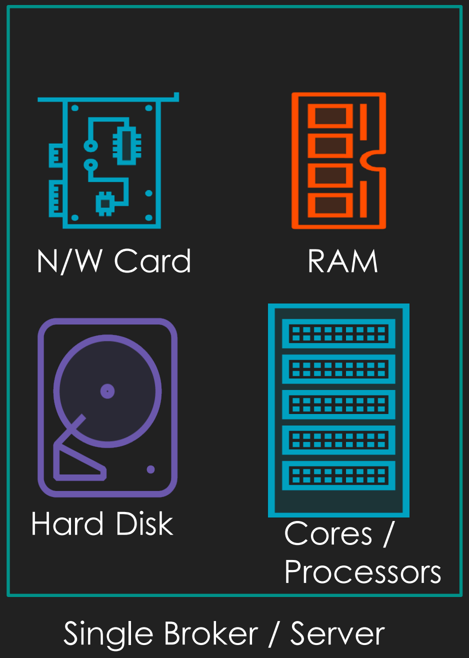
© 2019 Prudhvi Akella

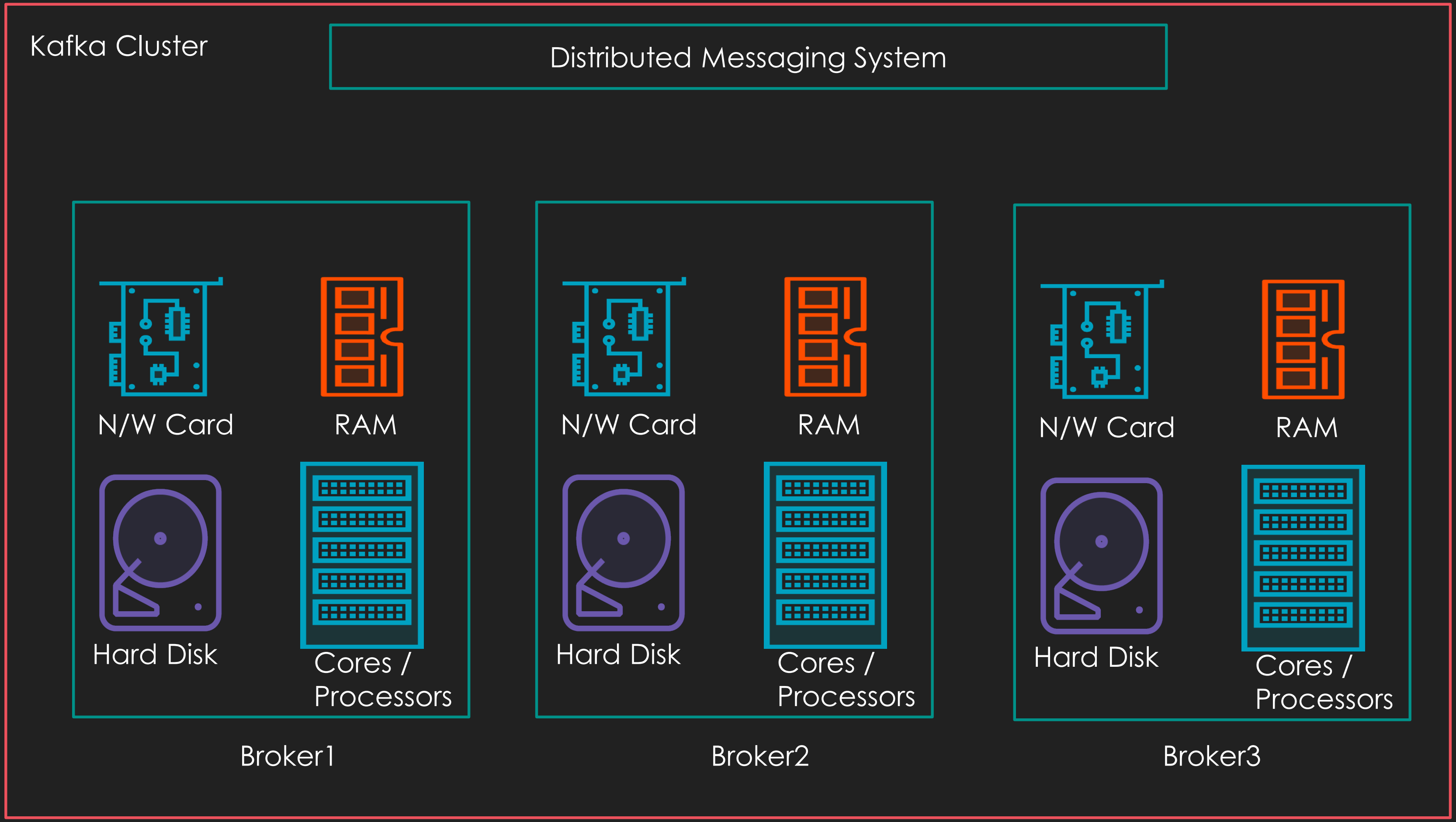
Major Components

- Brokers
- Zookeeper
- Topic
- Producer
- Consumers

© 2019 Prudhvi Akella

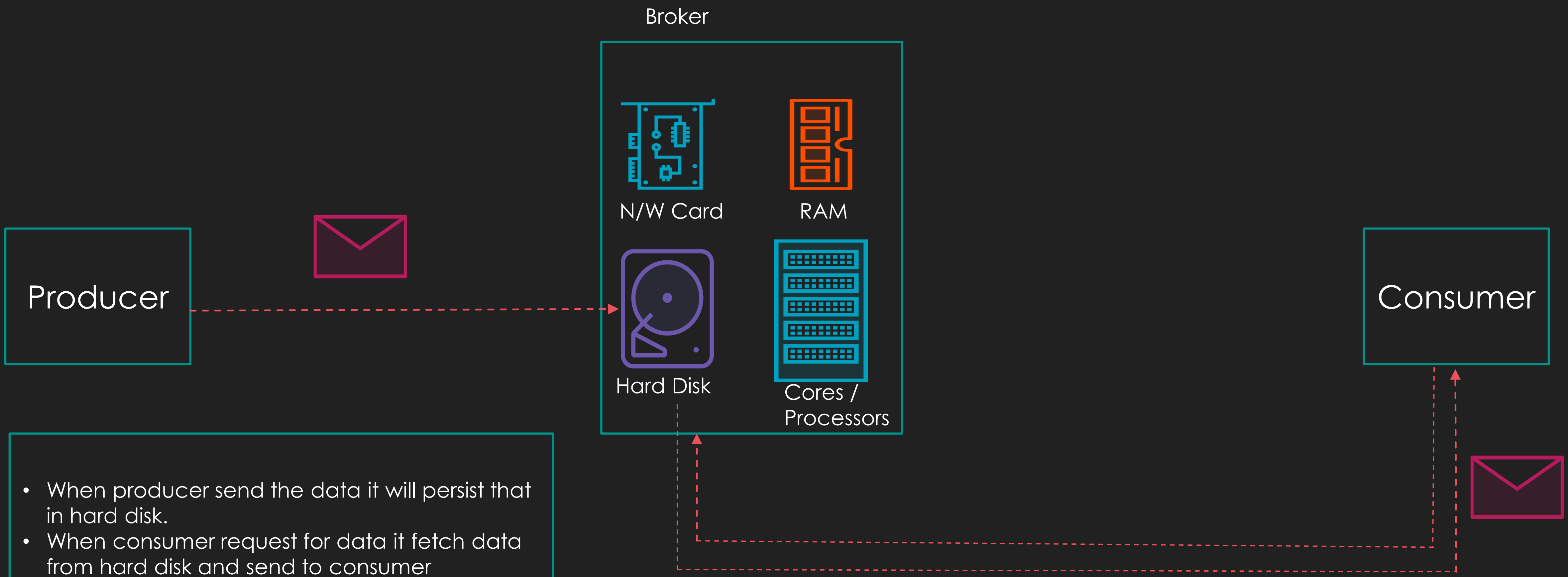
What is a Brokers/Bootstrap server





© 2019 Prudhvi Akella

What does a broker do?



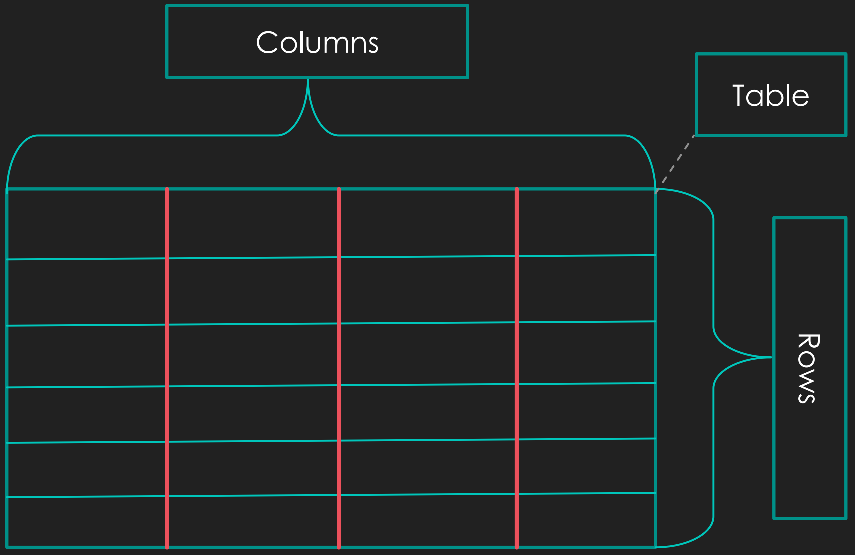
- When producer send the data it will persist that in hard disk.
- When consumer request for data it fetch data from hard disk and send to consumer

© 2019 Prudhvi Akella

How does broker store the Messages?

Topic

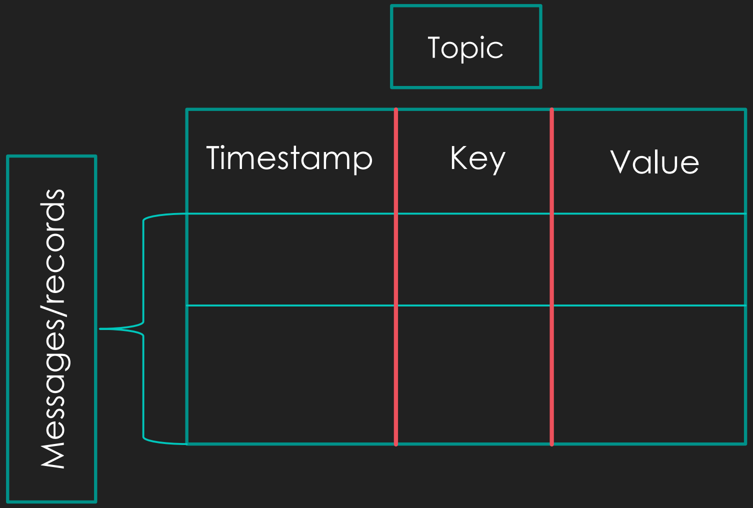
Before starting with Topic lets resume here and understand how table work in RDBMS?



- To store the data into RDBMS have to create a table
  With columns and there datatypes
- Once the table is create you can insert, update , select, delete data using table name as reference
- One row can be inserted into table at single point of time and that is considered to be a record
- When ever you do a bulk insert each row will be inserted in sequence(one after the other)

© 2019 Prudhvi Akella

Lets understand how topic works?



- To store the data into Kafka have to create a Topic
- Every topic in Kafka will have only 3 fields Timestamp, Key, Value
- When every a producer is sending records to cluster it has to include
- Four Things
  - 1)Key
  - 2)Value
  - 3)Topic Name
  - 4)Timestamp: Its Optional if producer add it will be used or else Kafka will add it while insertion
- If Consumer wants to get the records it has subscribe to topic while connecting to cluster.
- What is the Datatype of key, value?

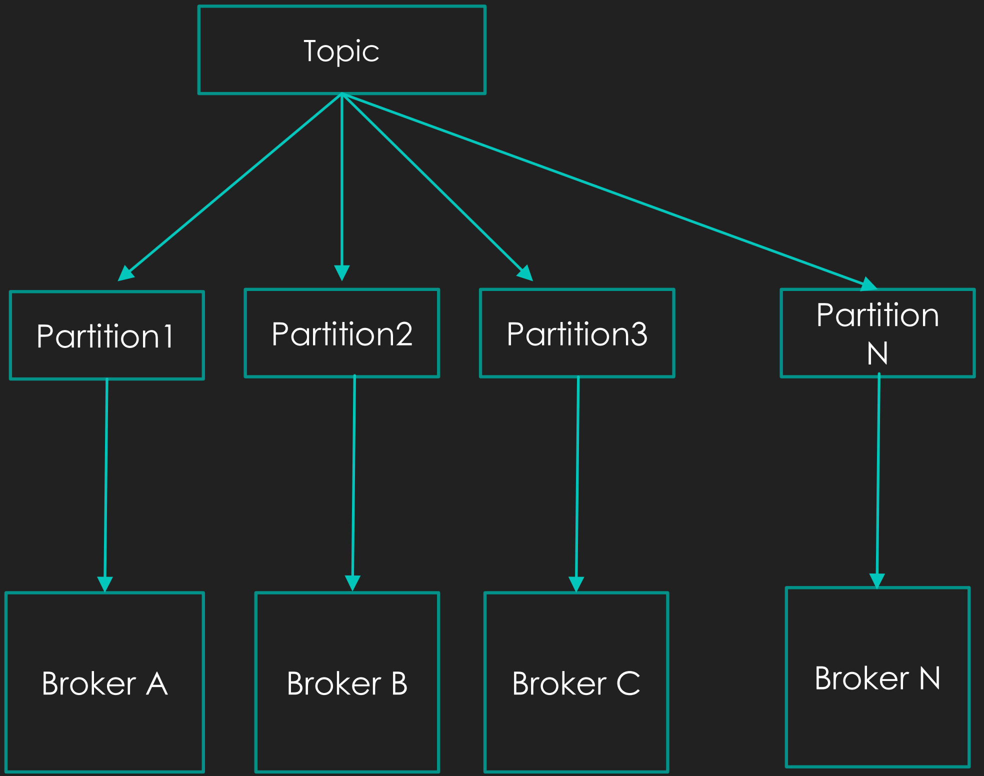
  Its Byte what ever the data stores in Kafka topic it will be in the form of bytes.

  We will discuss about this in detail when we talk about producers and consumers

© 2019 Prudhvi Akella

Topic Partitions

Kafka topic is divided into partitions and they are distributed across brokers so that cluster will be balanced. We can achieve parallel processing/distributed processing only when we have a distributed storage.

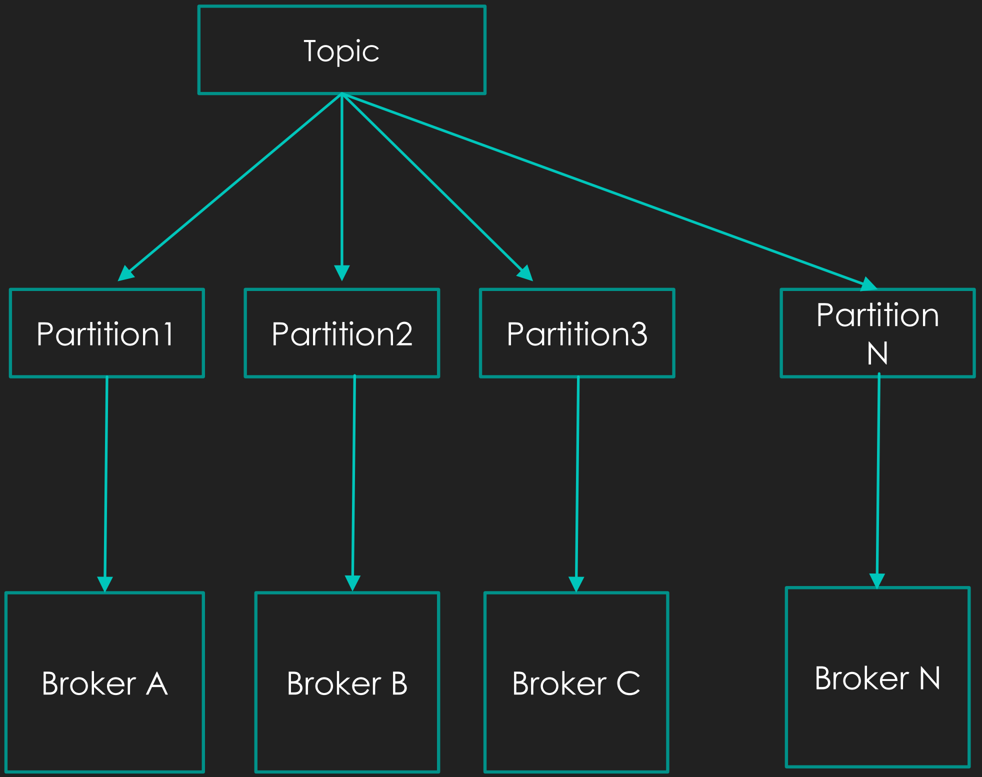


Small question? Lets say cluster contains 3 brokers and user created a topic with 4 partitions. Now how will be partitions distributed across brokers?

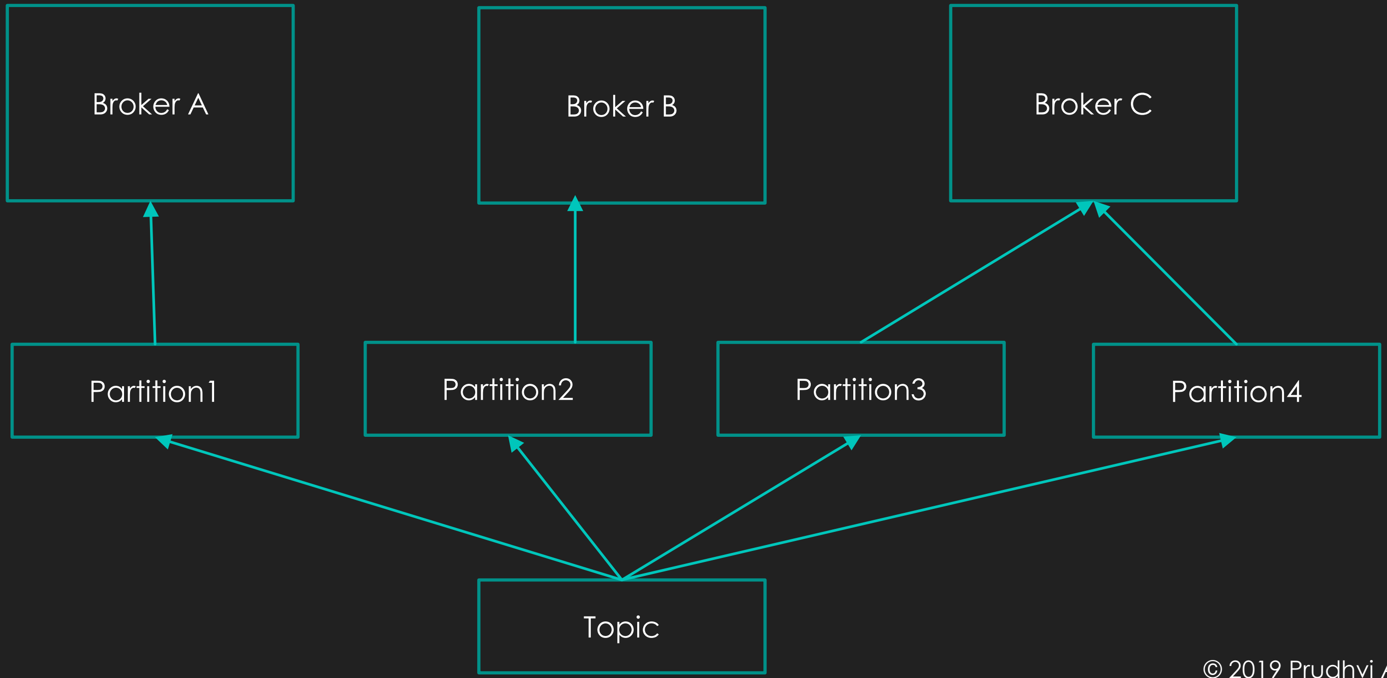
© 2019 Prudhvi Akella

Topic Partitions

Topic Partitions maintain rebalancing in cluster among broker Kafka topic is divided into partitions and partitions are distributed across brokers. This is a very common thing happens distributed system. We can achieve parallel processing/distributed processing only when we have a distributed storage



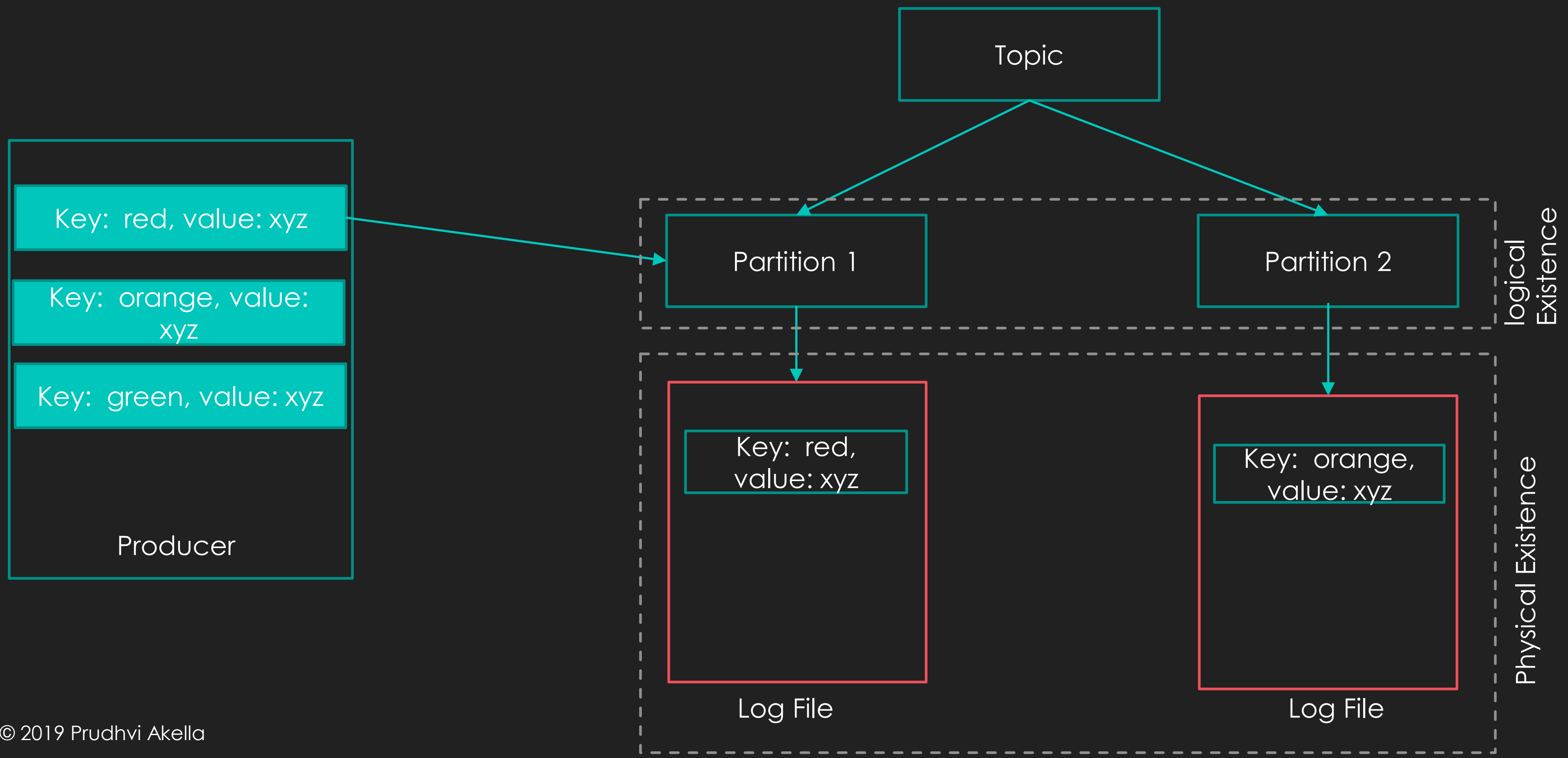
Small question? Lets say cluster contains 3 brokers and user created a topic with 4 partitions. Now how will be partitions distributed across brokers?



© 2019 Prudhvi Akella

What is Log File in Kafka?

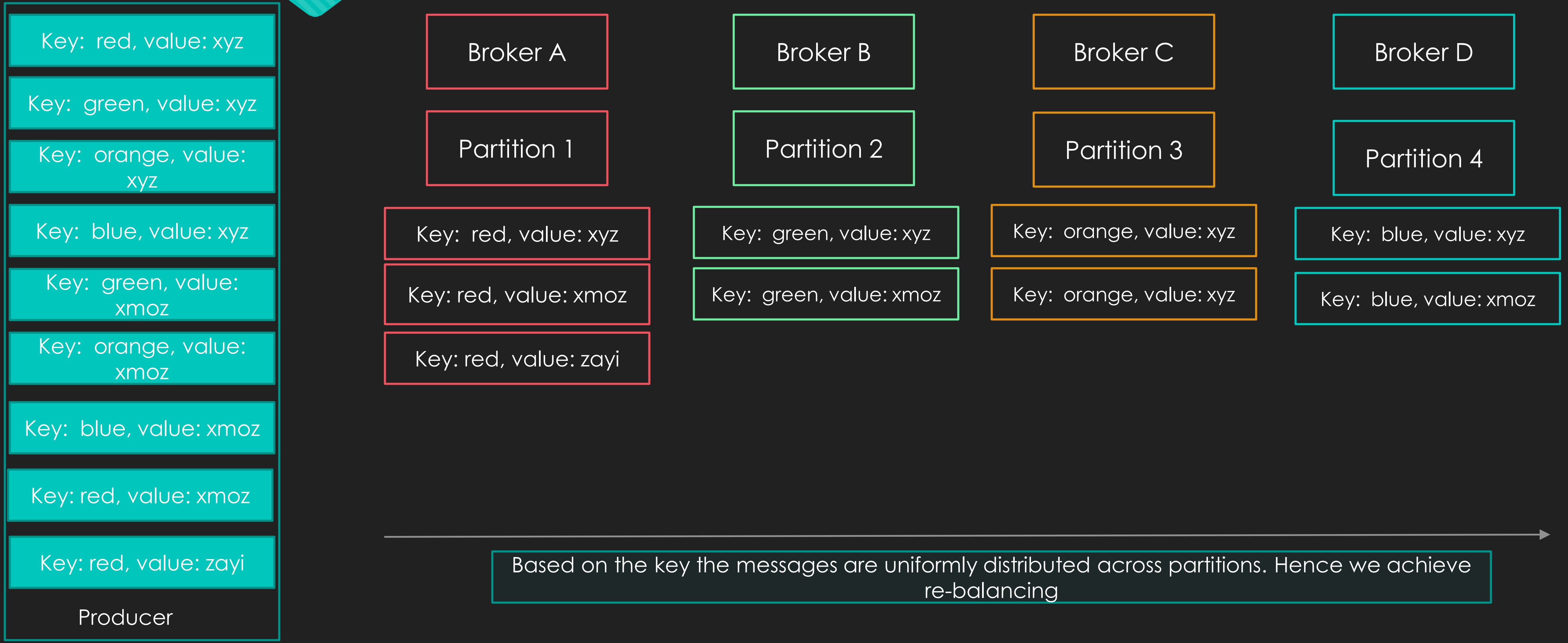
Log File is a place where messages will be stored physically and partition are called logical existence.



© 2019 Prudhvi Akella

How messages distributed across partitions in topic?

Assume that you have 4 brokers in cluster and user created a topic called colors with 4 partitions key of message will either of the one in red, green, blue, orange ?

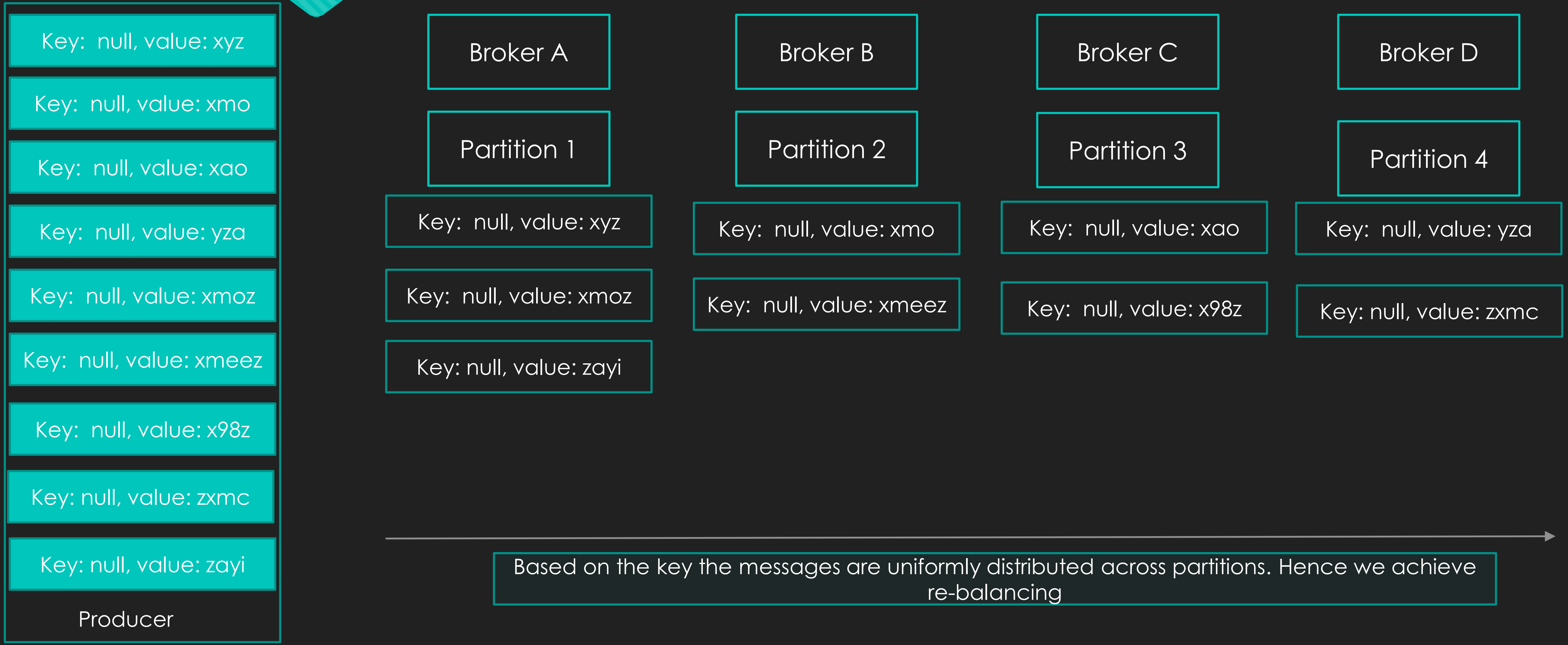


Based on the key the messages are uniformly distributed across partitions. Hence we achieve re-balancing

© 2019 Prudhvi Akella

How partitioning works if key=null?

Then Messages will be distributed in a **round robin fashion**



Based on the key the messages are uniformly distributed across partitions. Hence we achieve re-balancing

© 2019 Prudhvi Akella

How will messages get distributed across partitions?

Default Pratitioner is Murmur2 Hash Algorithm.

Murmur2 Algorithm hashs the key and puts the records into a particular partition and using a below formula

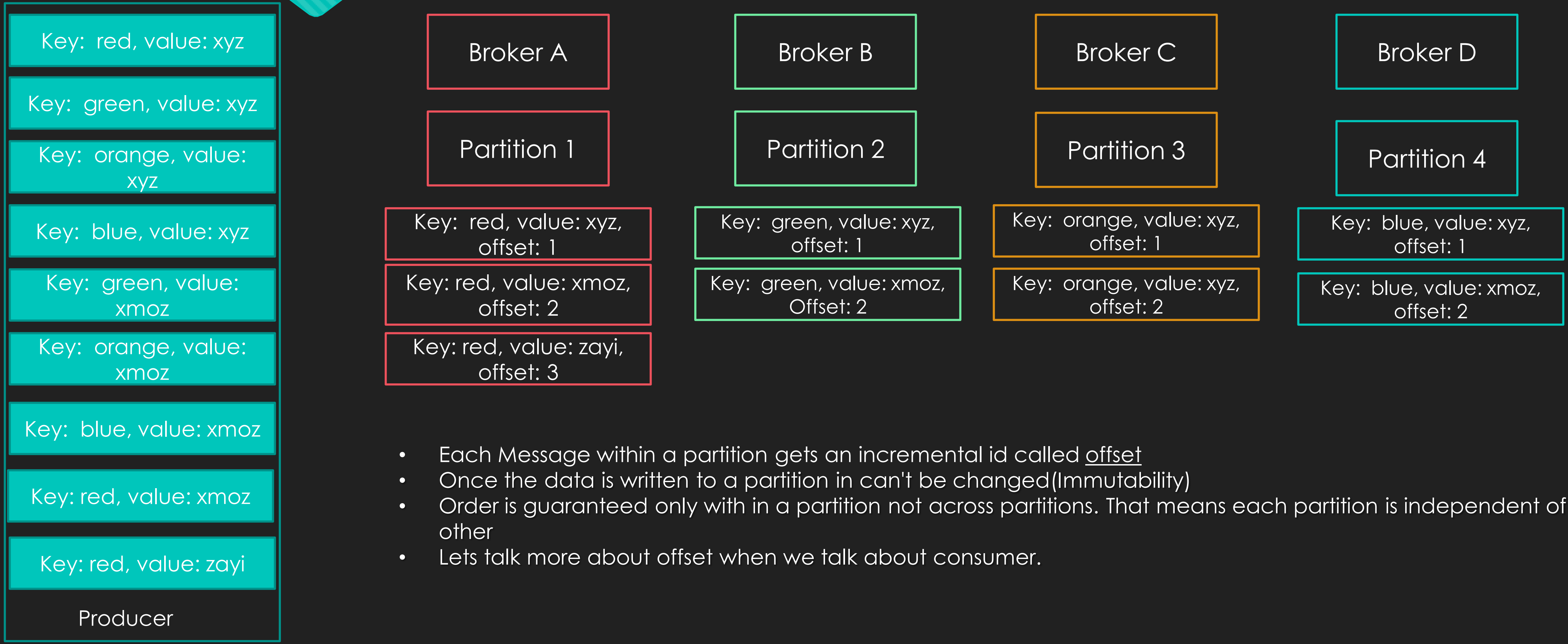
Targetpartition = Utils.abs(utils.murmur2(record.key) % numparitions

We can change the default behavior by overriding Partitioner class usually we wont do it.

When ever you create a topic you will mention the number of partitions

© 2019 Prudhvi Akella

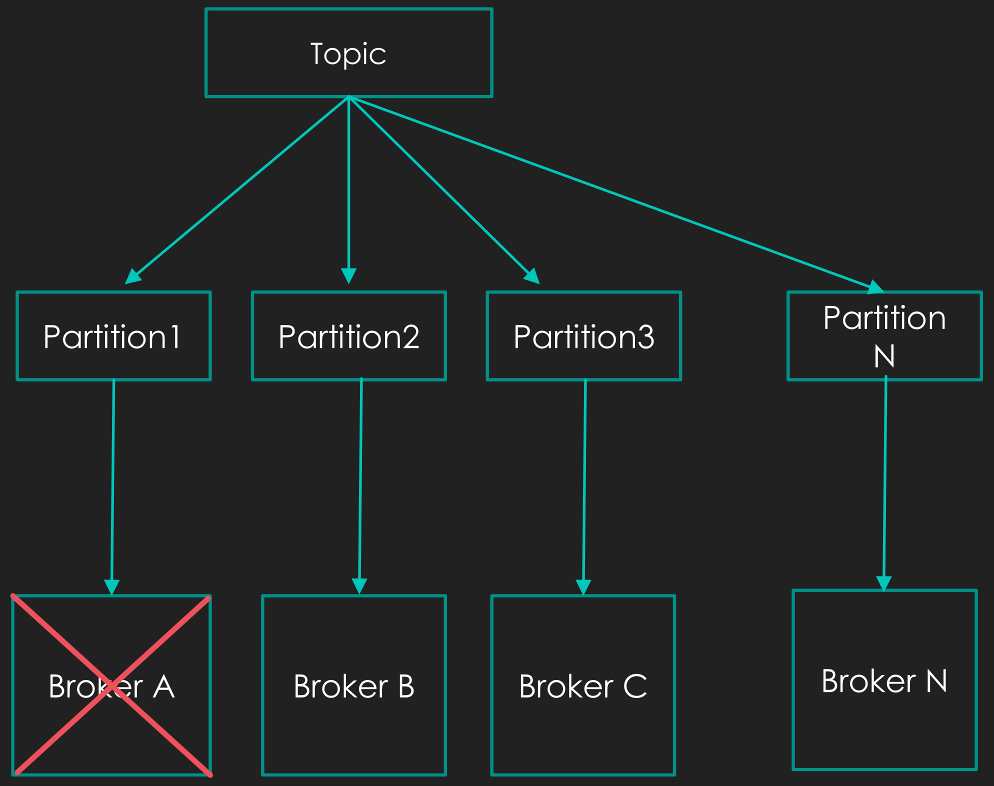
Offset


Key: red, value: xyz
Key: green, value: xyz
Key: orange, value: xyz
Key: blue, value: xyz
Key: green, value: xmoz
Key: orange, value: xmoz
Key: blue, value: xmoz
Key: red, value: xmoz
Key: red, value: zayi
Producer
Broker A
Partition 1
Key: red, value: xyz, offset: 1
Key: red, value: xmoz, offset: 2
Key: red, value: zayi, offset: 3
Broker B
Partition 2
Key: green, value: xyz, offset: 1
Key: green, value: xmoz, Offset: 2
Broker C
Partition 3
Key: orange, value: xyz, offset: 1
Key: orange, value: xyz, offset: 2
Broker D
Partition 4
Key: blue, value: xyz, offset: 1
Key: blue, value: xmoz, offset: 2

- Each Message within a partition gets an incremental id called offset
- Once the data is written to a partition in can't be changed(Immutability)
- Order is guaranteed only with in a partition not across partitions. That means each partition is independent of each other
- Lets talk more about offset when we talk about consumer.

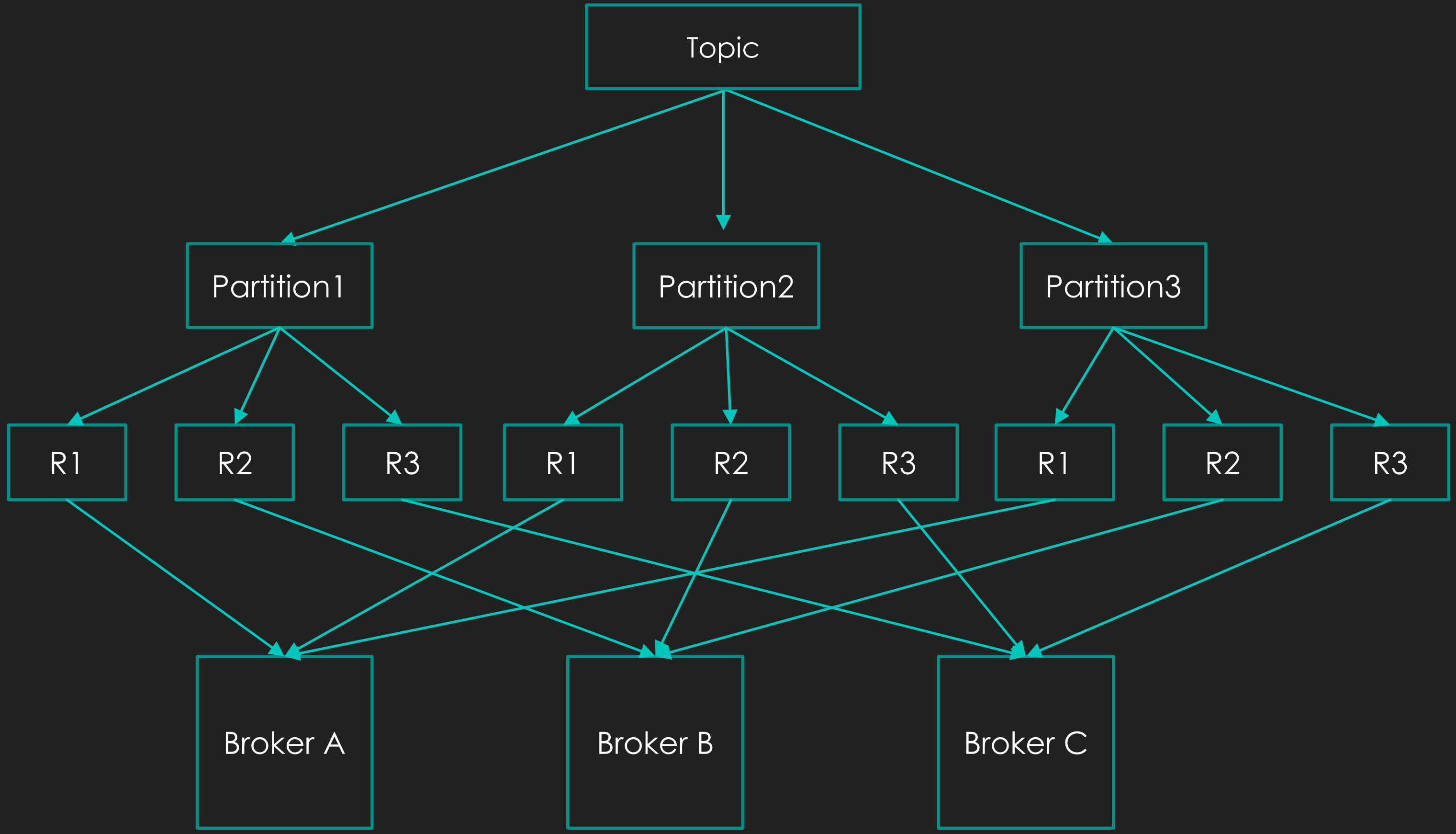
© 2019 Prudhvi Akella

What if Broker goes down?
What happens to partition data?



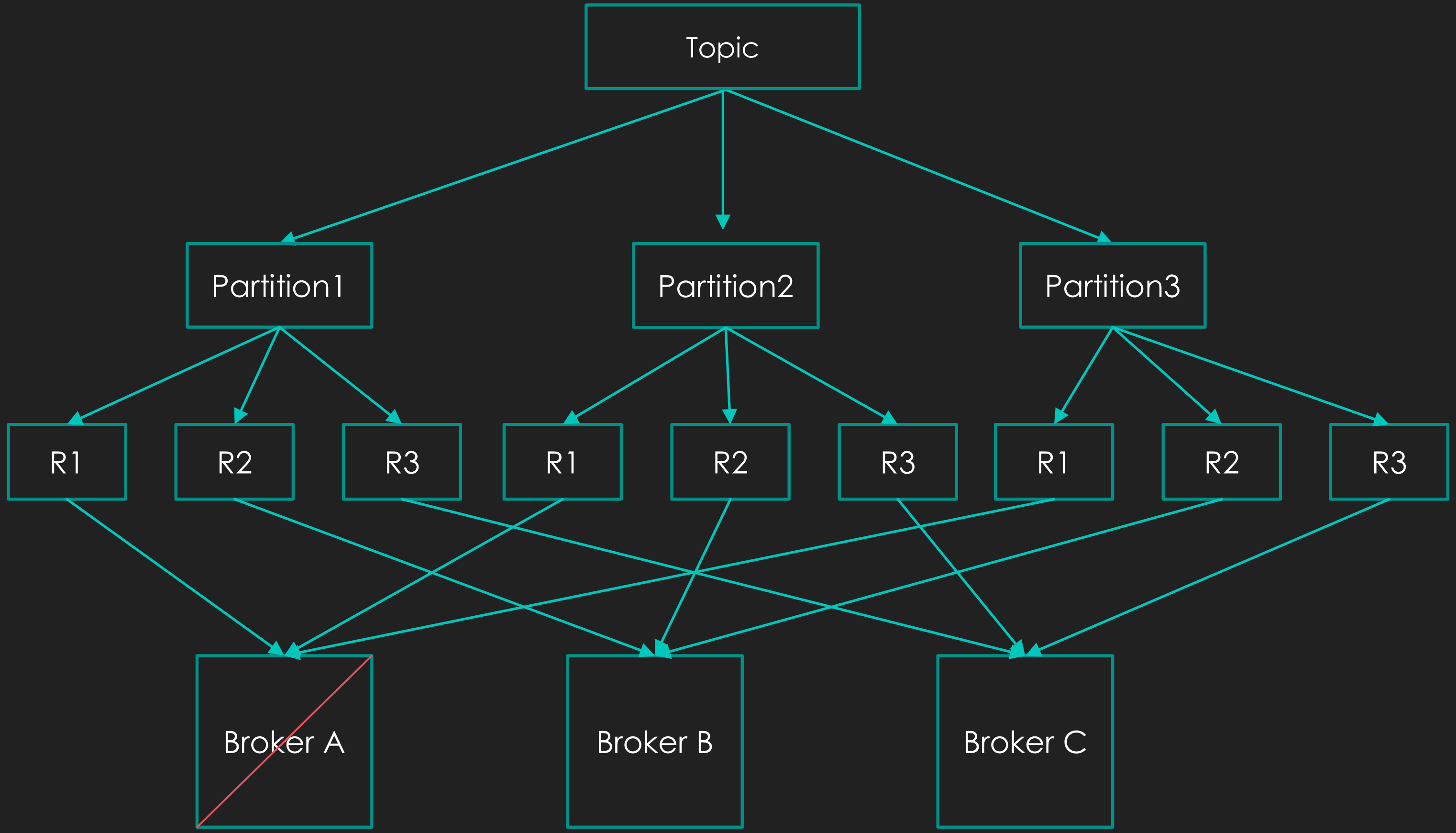
© 2019 Prudhvi Akella

Kafka has a plan to achieve fault tolerance. That is partition replication



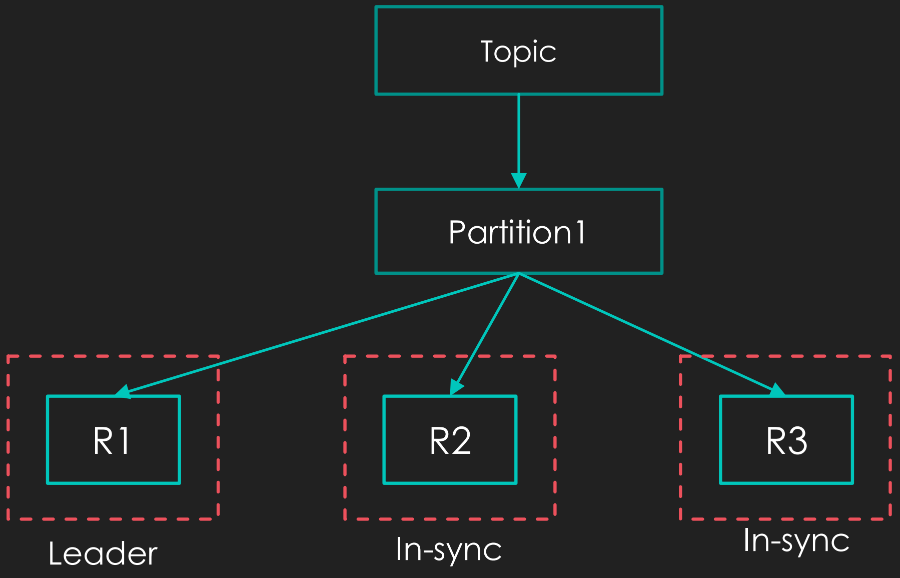
© 2019 Prudhvi Akella

Now if one broker goes down. The partition replica is available on other broker



© 2019 Prudhvi Akella

Leader and In-sync replicas



- Every topic partition will have replicas
- Out of replicas there will be one Leader and remaining are called In-sync replicas
- Zookeeper will conduct election b/w replicas and choose the leader out of it.
- We will discuss more about this when we talk about Producer Configuration

© 2019 Prudhvi Akella

Partition Count and Replica Count

- Its best to get particulars right at the first time at the time of topic creation.
- if partition count increases during a topic lifecycle of topic will break and keys ordering guarantees.
- If replication factor increases during a topic life cycle you put more pressure on your cluster which can lead to a unexpected performance decrease

Partition Count:

- Each Partition can handle a throughput of few MB/s
- More Partitions better performance and better throughput
- Ability to run more consumer groups at scale (This we will see when we talk about consumer and consumer groups)
- But more elections to perform for Zookeeper
- But more Logs file will open(Log files is a place where messages will store in partitions)

Small Cluster(<6 brokers) : count = 2 * Broker count
Big Clusters(> 12 Brokers): count = 1 * Broker count

Note: Partitions should be not more than 2000 to 40000 for broker and 20,000 per cluster because if broker goes down zookeeper has to perform lots of leader elections.

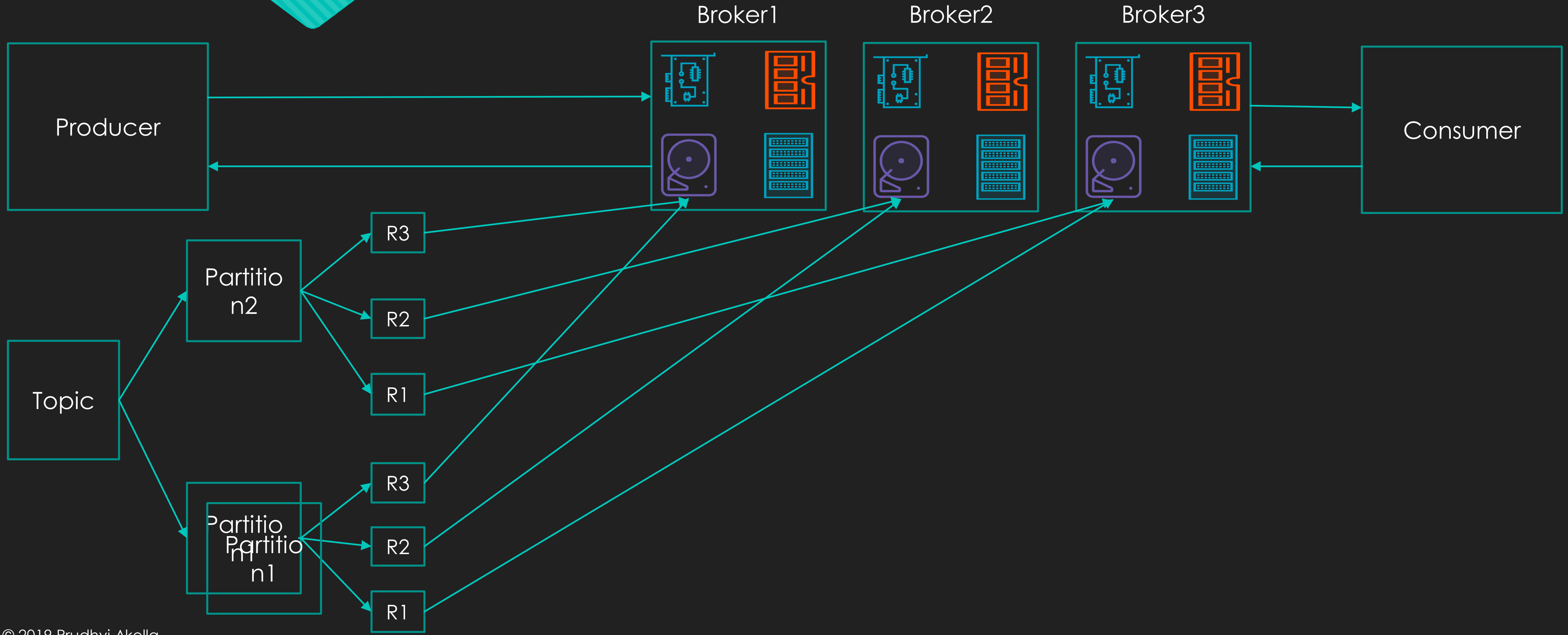
© 2019 Prudhvi Akella

Partition Count and Replica Count

- Replication Factor should be Atleast:2, Usually 3, Maximum: 4
- Better the resilience of your system N-1 brokers can fail
- In case of more replications(higher the latency if acks = all(we will discuss in details when we talk about producer).

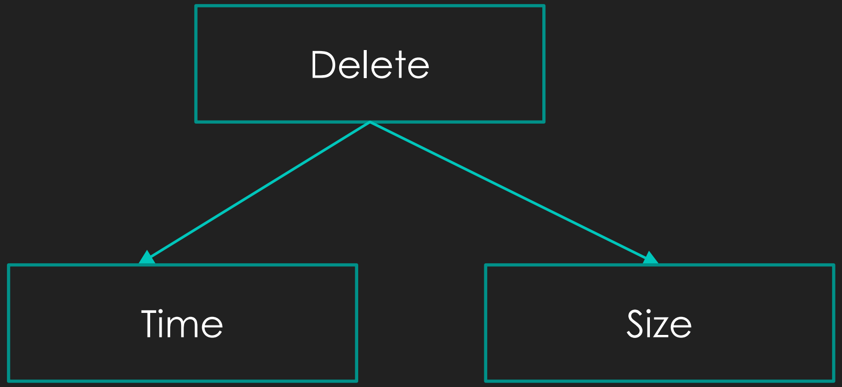
© 2019 Prudhvi Akella

Bird View



© 2019 Prudhvi Akella

How long data will reside in Topic?



Time:

- By Default broker is configured to delete the messages in 7 days.
- The property to set this property is log.retention.hours
- Lets say if you set your retention period to 1 day the message produced on day 1 will be deleted on day2

Size:

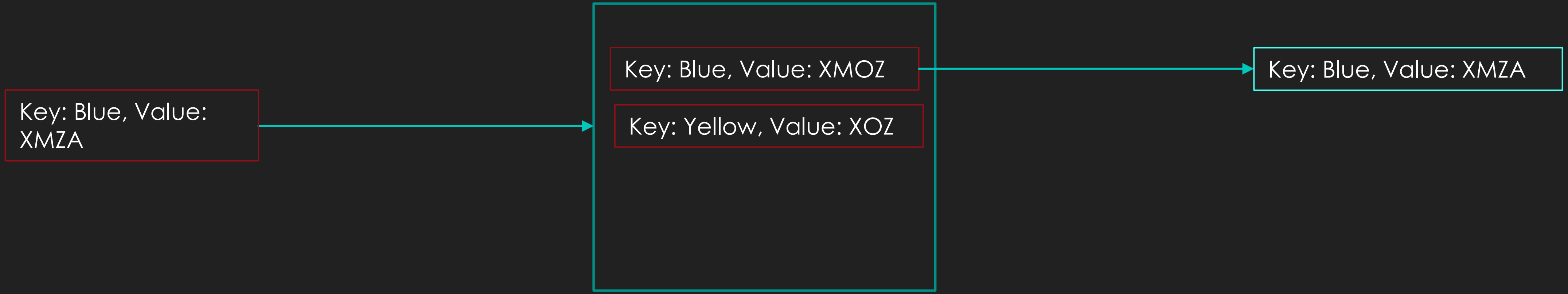
- The broker starts cleaning up the messages based on the space.
- Lets say maximum size for topic is set to 20KB and lets say each message has 5 kb so in your topic we can store max 4 messages. now lets say when 5 message arrives the system the old ones are deleted.
- By default no value will be set in configuration .
- The Property to set size is log.retention.bytes

© 2019 Prudhvi Akella

How long data will reside in Topic?

Compaction

Compaction in Kafka works as Upsert(Update + Insert).That means when a new message is produced by the producer to broker then broker check whether record with key exists or not if exist it updates the value and if it is not will insert the value



© 2019 Prudhvi Akella

What is Zookeeper?

Zookeeper provides multiple features for distributed applications:

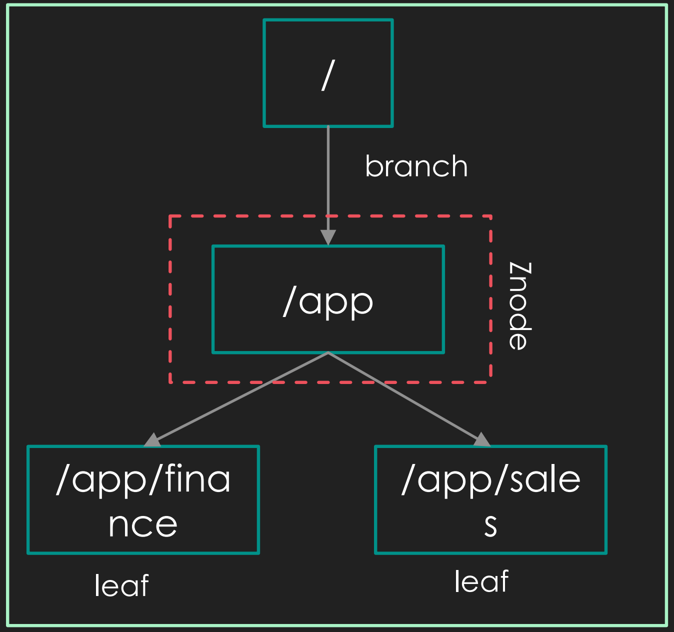
  - Distributed configuration management
  - Self election
  - Coordination and locks(low level)
  - Key value store

- Zookeeper used in many distributed systems such as Hadoop, Kafka, Hbase etc.
- Its an apache project that's proven to be very stable and hasn't had a major release in many years
- 3.4.x stable version
- 3.5.x is in development for many years, and its still beta(not for production use)

© 2019 Prudhvi Akella

What is Zookeeper?

We all know how Linux file system looks like its starts from / folder then extended by different directories
Example /home/ec2-user/
zookeeper also looks in the same way



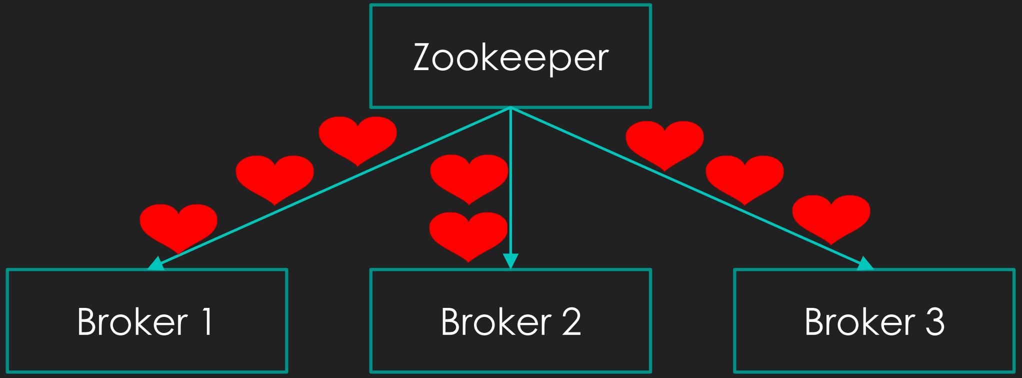
Tree

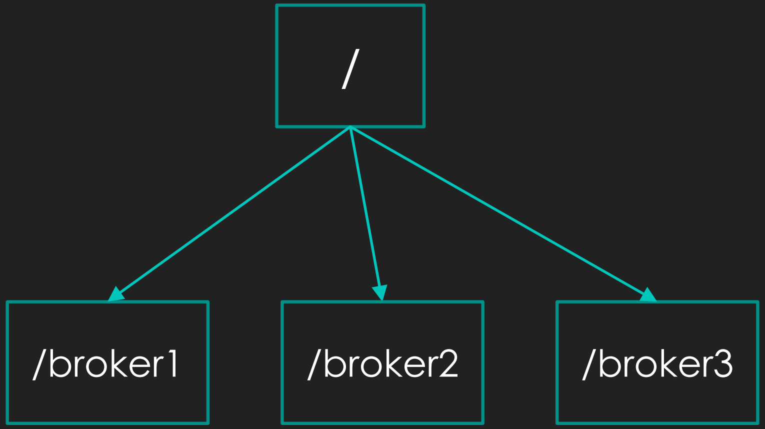
- Zookeeper Internal data structure structure is like Tree. It has leafs and branch's.
- Each node is called a zNode.
- Each node has a path
- Each node can be persistence or ephemeral. what the difference?
- persistence zNode will alive all the time.
- ephemeral zNode go away if your app disconnect
- Each Znode can store multiple zNode or it can store data.
- We cannot rename zNode.
- One of the best feature of zookeeper is it Watched for Changes. say if any change is occurred in /app/finance it will let me know hey hey there is some change in /app/finance check it out.

© 2019 Prudhvi Akella

Role of Zookeeper?

Broker Registration, with heart beat mechanism to keep the list of current.
When broker registration happens a zNode will be created.





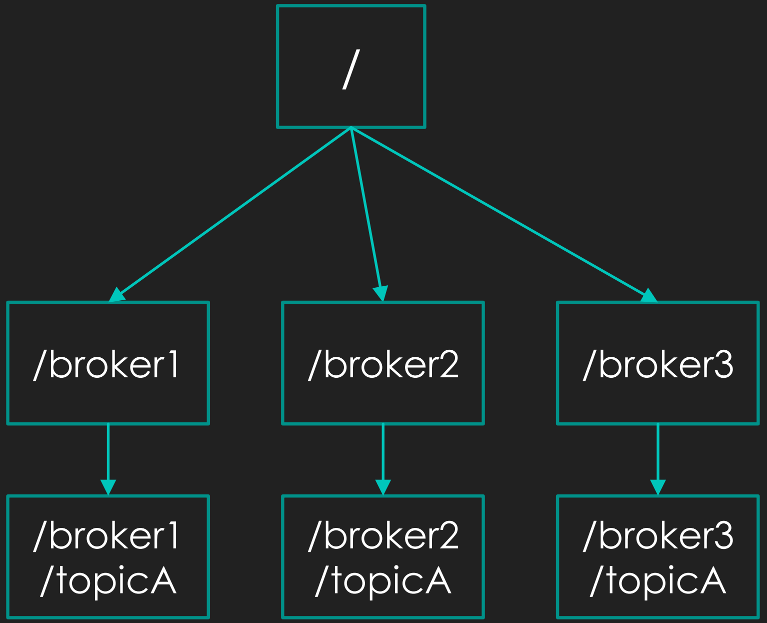
© 2019 Prudhvi Akella

Role of Zookeeper?

Maintaining the list of topic alongside(when ever a topic is created it create a zNode in zookeeper all the info for topic will be stored there)

- Their configurations(Partitions, replication factor, additional configurations)
- The list of ISR(Insync replicas) for partitions

Performing leader elections in case of broker goes down



© 2019 Prudhvi Akella

Role of Zookeeper?

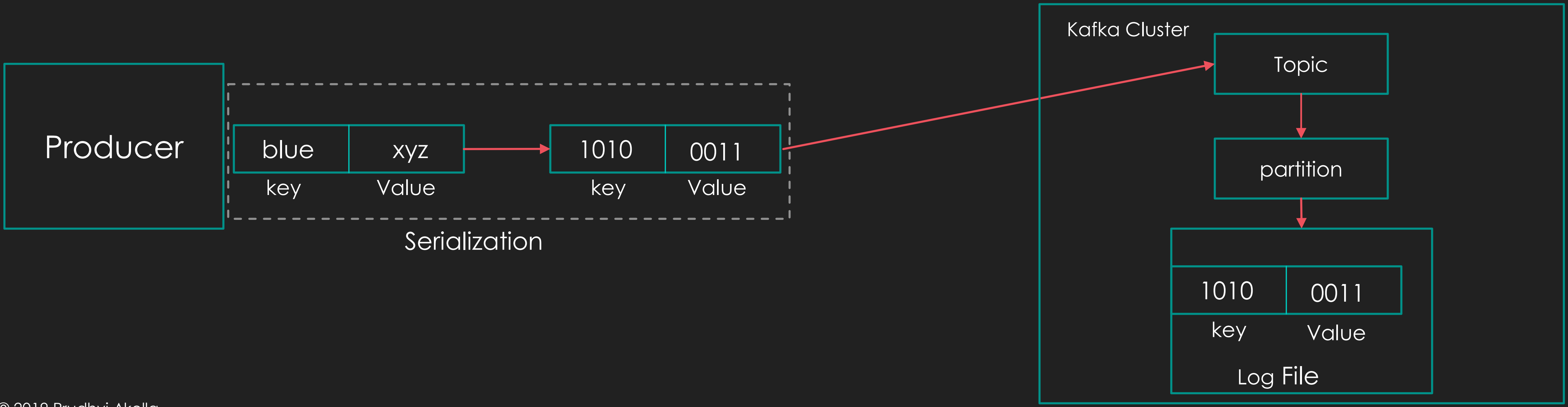
- Storing the Kafka Cluster ID(randomly generated at 1st startup of cluster)
- Storing ACL's(Access control list) if security is enabled
-     Topics
-     Consumer groups
-     user
- (Depreciated: Not in use) Used by old consumer API to store offsets.

© 2019 Prudhvi Akella

Serialization

The process of transforming object to byte is called serialization. Kafka cluster/topic can store only bytes so when producer is sending the messages to topic messages(Key, Value) has to be serialized or converted to bytes. Conversion process will happen at producer end using serializes

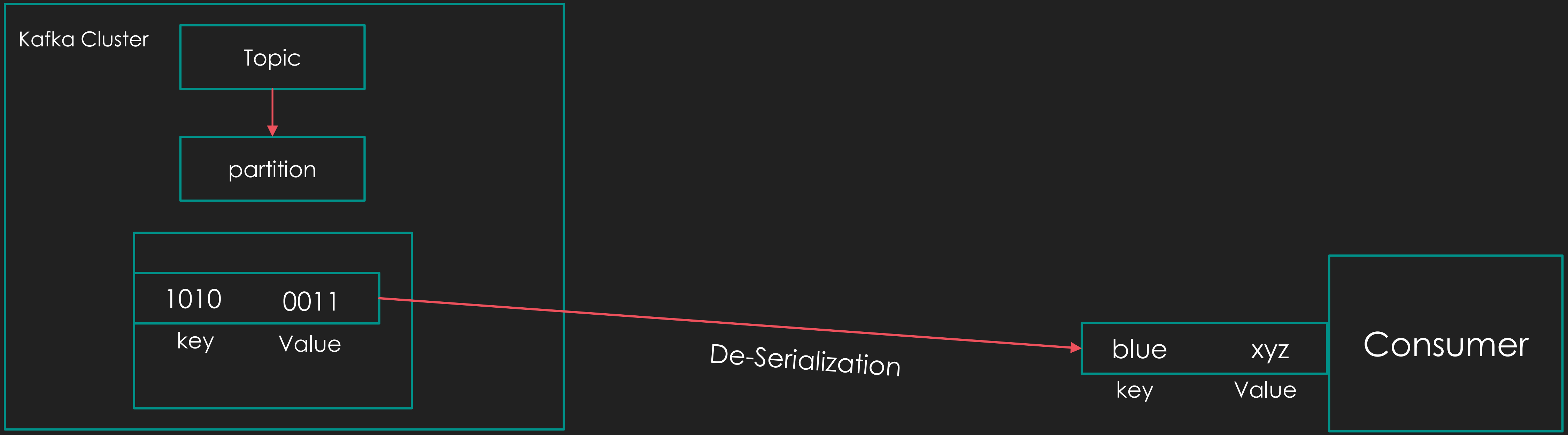
Default serializers provided by Kafka are String, Long, Int. for custom object serialization we have to depend on AVRO Serialization we will talk in detail about this.



© 2019 Prudhvi Akella

De-Serialization

The process of transforming byte to object is called de-serialization. When Consumer connects to the Kafka and subscribe for a topic then Kafka send messages in bytes which has to be de-serialized back to message at consumer for further processing.



© 2019 Prudhvi Akella

Enough of Theory lets make your hands dirty.

© 2019 Prudhvi Akella

Lets setup Kafka

Prerequisite :

- ✓ Java8
- ✓ kafka_2.12-2.3.0
- ✓ IntelliJ IDEA
- ✓ Scala
- ✓ SBT(Scala Build Tool).

© 2019 Prudhvi Akella

First Demo

- Configure Zookeeper and Kafka.
- Start Zookeeper and Kafka.
- Create a topic and list the topics
- Start Kafka Console Producer and Console Consumer.
- Send message from Producer it has to reach the consume.

© 2019 Prudhvi Akella

First Demo

Start Zookeeper:
- Go to Kafka/bin/windows
- Run zookeeper-server-start.bat Kafka config\zookeeper.properties

Start Kafka
- Go to kafka/bin/windows
- Run kafka-server-start kafka.bat config\server.properties
- wait until Kafka start.

Create Topic:
- Go to kafka\bin folder
- Run

kafka-topics.bat kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic First-Producer

Topic List:

kafka-topics.bat --list --bootstrap-server localhost:9092

© 2019 Prudhvi Akella

First Demo

<u>Kafka Console Producer:</u>

```
kafka-console-producer.bat --broker-list localhost:9092 --topic First-Producer
```

<u>Kafka Console Consumer:</u>

```
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic First-Producer --from-beginning --formatter
kafka.tools.DefaultMessageFormatter --property print.key=true --property print.value=true --property
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer --property
value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

Now type message in producer console and click enter you must able to see it in consumer console.
Note:
Now navigate to tmp\kafka_logs and check folder with topic name has to be created in that you will find 00000000.log file in that you will see all the messages.

© 2019 Prudhvi Akella

Demo-Scala Producer

- ✓ Configure build.sbt file by adding necessary Kafka libs
- ✓ Develop producer with configuration
- ✓ Start Kafka Console Consumer
- ✓ Start pushing messages from producer. Consumer should receive them.

© 2019 Prudhvi Akella

Producer Configuration

awks:

0: Possibility of data loss is very high no acknowledgement from Leader or In-sync replicas.

1: Possibility of data loss is moderate Leader will send the acknowledgement to producer once the messages is received.

All: Possibility of data loss is very less because both leader and In-sync replicas has to acknowledgement to producer.

min.insync.replicas:

This can be set either in broker level(applicable to all topics) or topic level

If this property is set to 2 and awks = All then at every point of time min brokers has to be available = 2 or else it will throw an exception

© 2019 Prudhvi Akella

Producer Configuration

<u>retries:</u>

In case of N/W or Hardware failures the developer has to handle the exceptions otherwise there will loss of data. if we set retries property producer will be keep retrying until cluster comes up.

By default this property is set to 0 for zero data loss set it to Interger.Max_Value

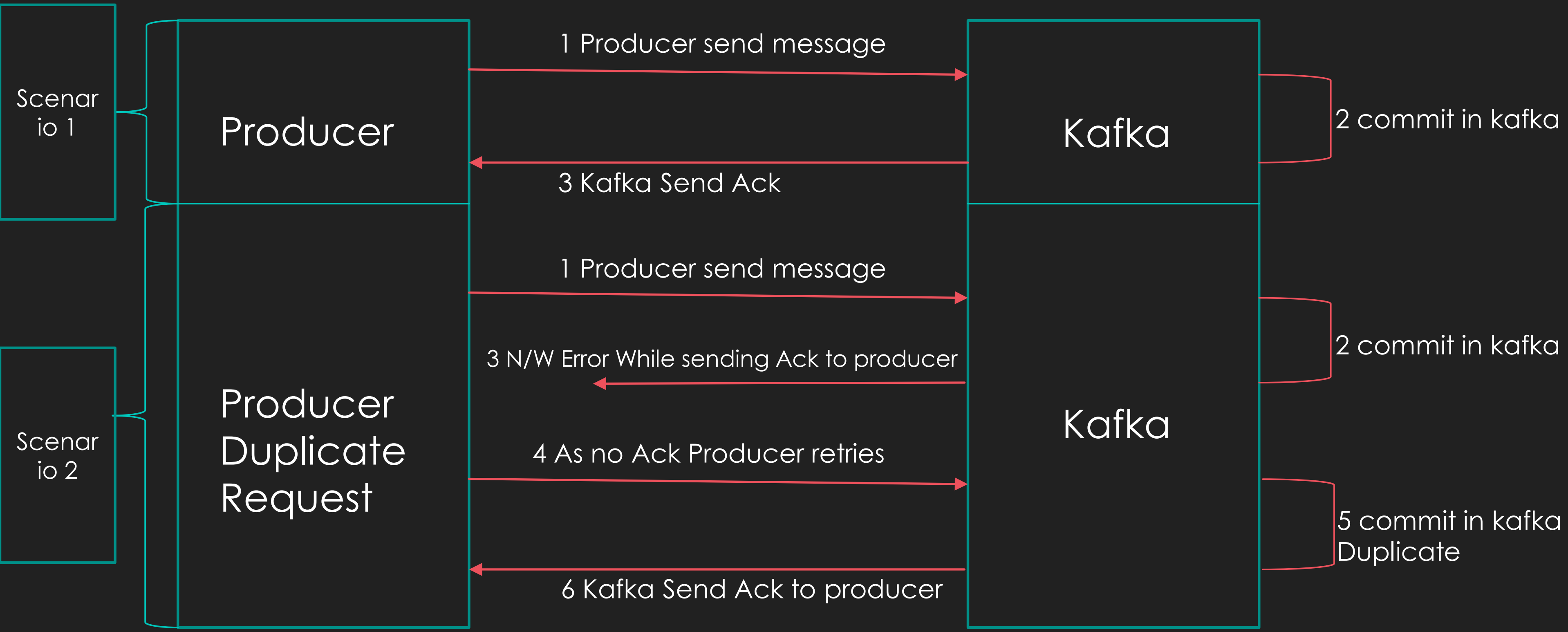
max.in.flight.request.per.connection:

In case of more retries there is a possibility of messages out of order that means messages will not send in a proper order one after another. for that reason if messages has to go in a proper order have to set this property. Set this property to 5 for proper ordering and high performance.

© 2019 Prudhvi Akella

Producer Configuration

What is the solution?



Scenar io 1
Producer
1 Producer send message
Kafka
2 commit in kafka
3 Kafka Send Ack
Scenar io 2
Producer Duplicate Request
1 Producer send message
2 commit in kafka
3 N/W Error While sending Ack to producer
Kafka
4 As no Ack Producer retries
5 commit in kafka Duplicate
6 Kafka Send Ack to producer

© 2019 Prudhvi Akella

Producer Configuration

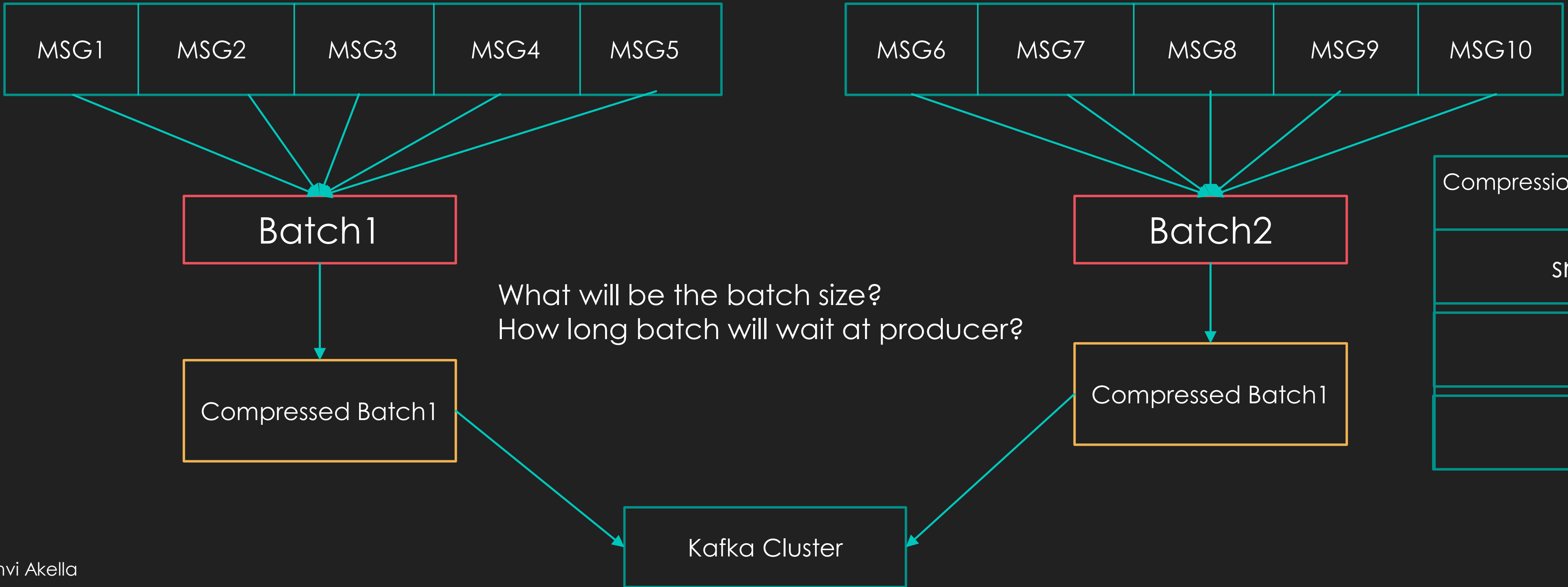
Solution: Idempotent Producer
Property : enable.idompatance = true

Now if producer is Idempotent there will no chance of duplicate commits because when there is a retry producer request it will append request ID to message it checks whether it is already committed or not with that id if it is already committed it will not commit again.

© 2019 Prudhvi Akella

Producer Configuration: How will decide throughput and latency?

Compressing the batch of messages is one of the optimization used in kafka to Increase the Throughput.
Property : compression= snappy



What will be the batch size?
How long batch will wait at producer?

Compression supported
snappy
bzip
lz4

© 2019 Prudhvi Akella

Producer Configuration

Batch.size:

Max number of bytes that will included in batch. The default is 16KB.

Increasing batch size to 32Kb or 64 Kb can help increasing the compression, throughput and efficiency of requests.

Linger.ms:

By default kafka tries to minimize the latency that means as soon as the message is received the kafka sends the message to cluster.

To Change this behavior and make kafka wait for a while to form a batch linger.ms is used to increase the throughput while maintaining the low latency.

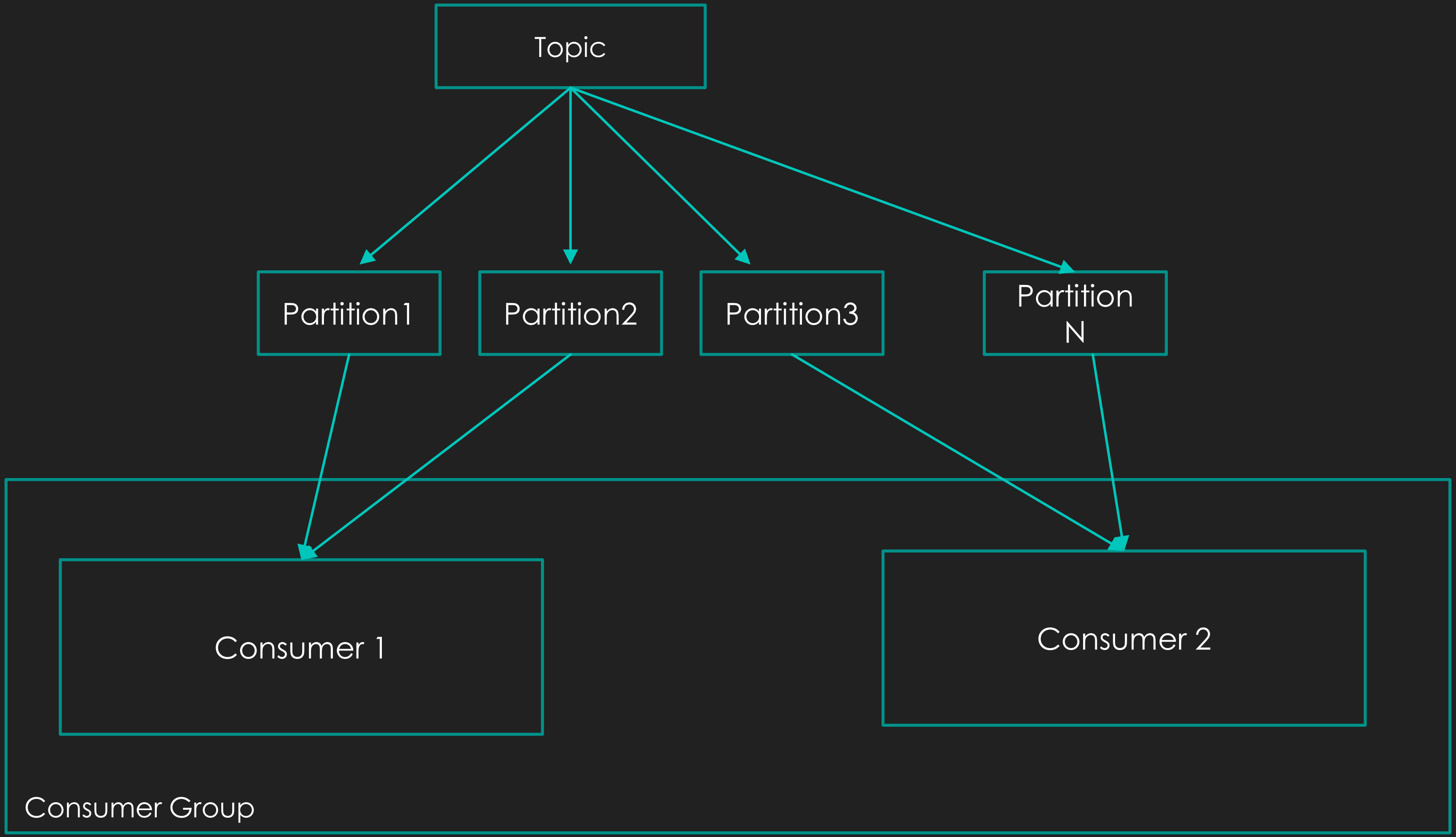
Linger.ms = Number of milliseconds a producer is going to wait to send the batch by default its set to 0.

Introducing the little delay will increases the throughput, compression and efficiency of a producer.

If the batch is full before the end of linger.ms period it will send kafka right away.

© 2019 Prudhvi Akella

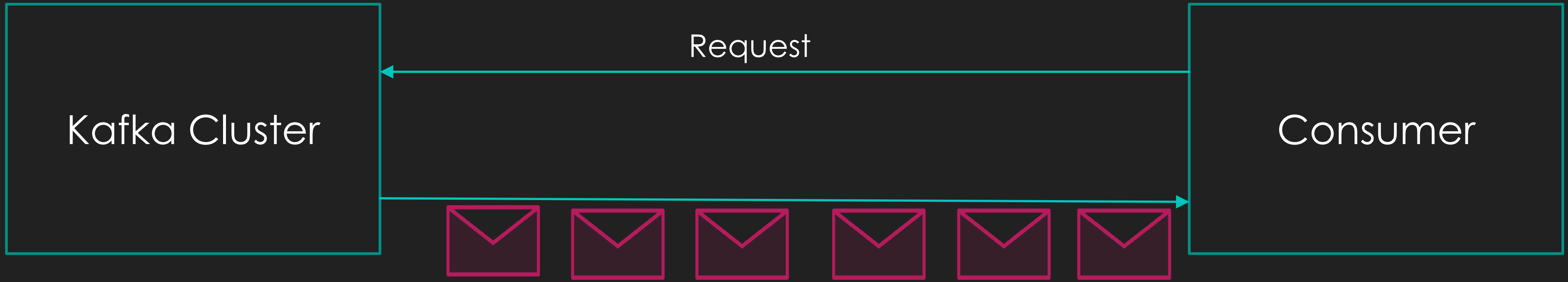
Consumer Configuration: Consumer Group



© 2019 Prudhvi Akella

Consumer Configuration: Poll

Poll is used to get messages from the kafka. lets say if it is set to 100ms at every 100ms of time consumer will request the kafka for messages that is called fetch. If no messages are available then it returns null.



You can control the poll data:

1) fetch.min.bytes: how much data you want to pull at least on each request. Default 1 MB
2) fetch.max.bytes: Max data returned for each fetch request. Default 500 Mb
3) Max.partition.fetch.bytes: Max data returned by broker per partition. Default 1MB
4) Max.poll.records: how man records to receive per poll request. Default 500

© 2019 Prudhvi Akella

Consumer Delivery Schematics

Kafka stores offsets at which a consumer group has been reading. these offsets are committed live in kafka topic name _consumer_offsets. In case of consumer dies it will be able to read back from where it left off thanks to committed consumer offsets. Offset commit depends on the schematics that you are choosing.

Atmost Once: Offsets are committed as soon as messages is received, If processing goes wrong the message will be lost it wont read again. It is not preferred.

Atleast Once: Offsets are committed only if message is processed at consumer side. If processing goes wrong the messages will read again there is a chance of duplication.so we have to make consumer idempotent. It is usually preferred.

Exactly Once : This can be achieved by kafka work flow using stream API's. Even in case of any failures record will be processed only once. No chance of duplication here.

© 2019 Prudhvi Akella

Consumer Configuration: Offset commit

enable.auto.commit:

If this property is set to true the moment the message is processed offset will be committed. By this we can achieve Atleast Once behavior.

If it is set to false It means manually user has to commit the offset using sync() method which is not recommended in production

© 2019 Prudhvi Akella

Lets Start

- What is Kafka?
- How Kafka works?
- Why Kafka?

© 2019 Prudhvi Akella

- Community Support.
- Java was initially started as Scala project in LinkedIn then later it is converted to JAVA project because java one of the commonly used across the globe. Then later kafka application can built in Scala, Java, Python.
- High Throughput: Once the data is serialized at producer and moved to kafka the messages will easily replicated across the replicas because bytes transfer requires very low bandwidth so throughput increases.
- Zero Copy: Usually in java applications once the data is received in consumer end through network cards they have to go through java heap(RAM) and then the data will be moved to hard disk but in kafka Java Heap is bypassed and messaged will directly store in hard disk. But this feature is available only if you disable TLS(Transport layer security) if you enable it every message has to go through Java Heap.

© 2019 Prudhvi Akella