Apache Kafka

HornetQ

ActiveMQ

Why Kafka?

When we have….
Aren't they Good?

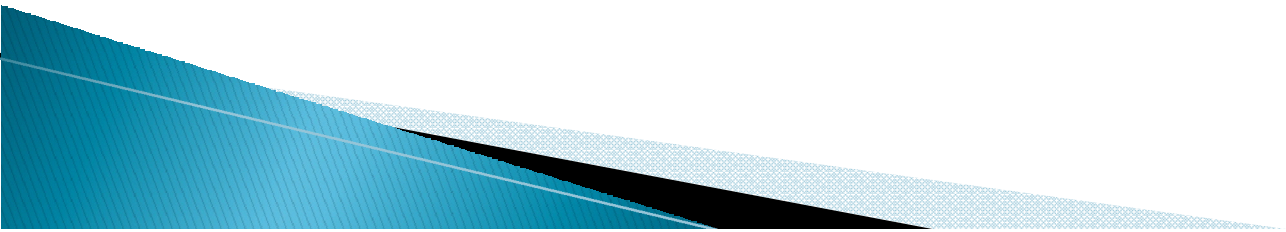
ØMQ

RabbitM

Open Source Enterprise Mess

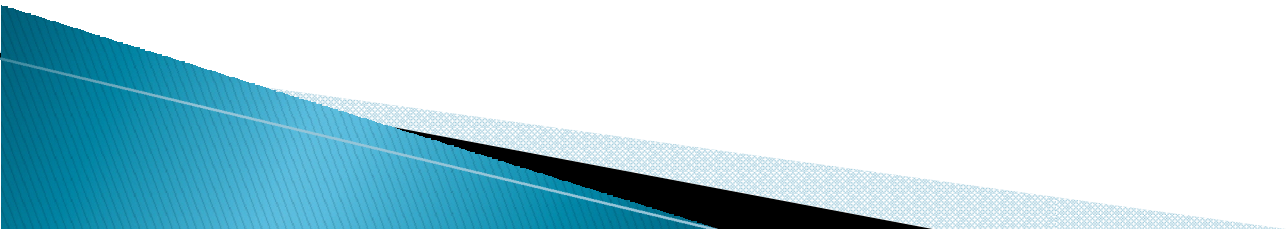
*Apache ActiveMQ, JBoss HornetQ, Zero MQ, RabbitMQ are respective brands of Apache Software Foundation, JBoss Inc, iMatix Corporat
Vmware Inc.

They all are Good
But not for all use-cases.



Use Cases

- Real-Time Stream Processing (combined with Spark Streaming)
- General purpose Message Bus
- Collecting User Activity Data
- Collecting Operational Metrics from applications, servers or device
- Log Aggregation
- Change Data Capture
- Commit Log for distributed systems



What is Common?

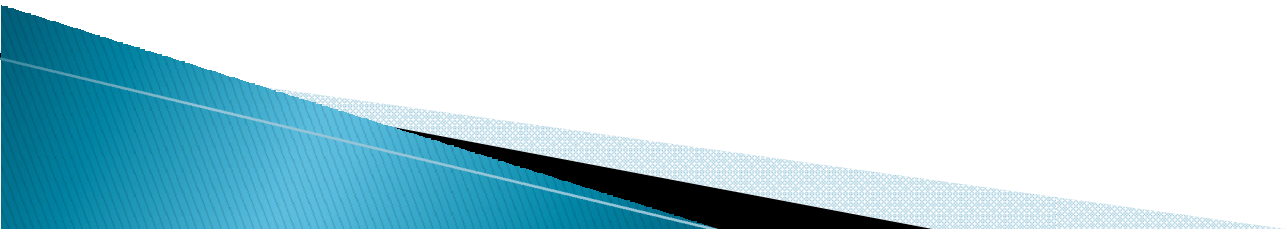
Scalable: Need to be Highly Scalable. A **lot** of Data. It can be billions of message.

Reliability of messages, What If, I loose a small no. of messages. Is it **fine** with me ?.

Distributed : Multiple Producers, Multiple Consumers

High-throughput: Does not require to have **JMS** Standards, as it may be overkill fo some use-cases like transportation of logs.

- As per JMS, each message has to be **acknowledged** back.
- Exactly one delivery guarantee requires **two-phase commit**.



Introduction

An Apache Project, initially developed by LinkedIn's SNA tear

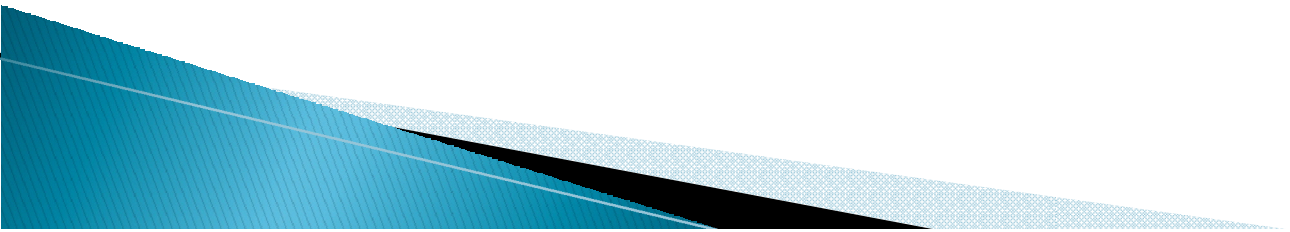
A High-throughput distributed Publish-Subscribe based messaging system.

A Kind of Data Pipeline

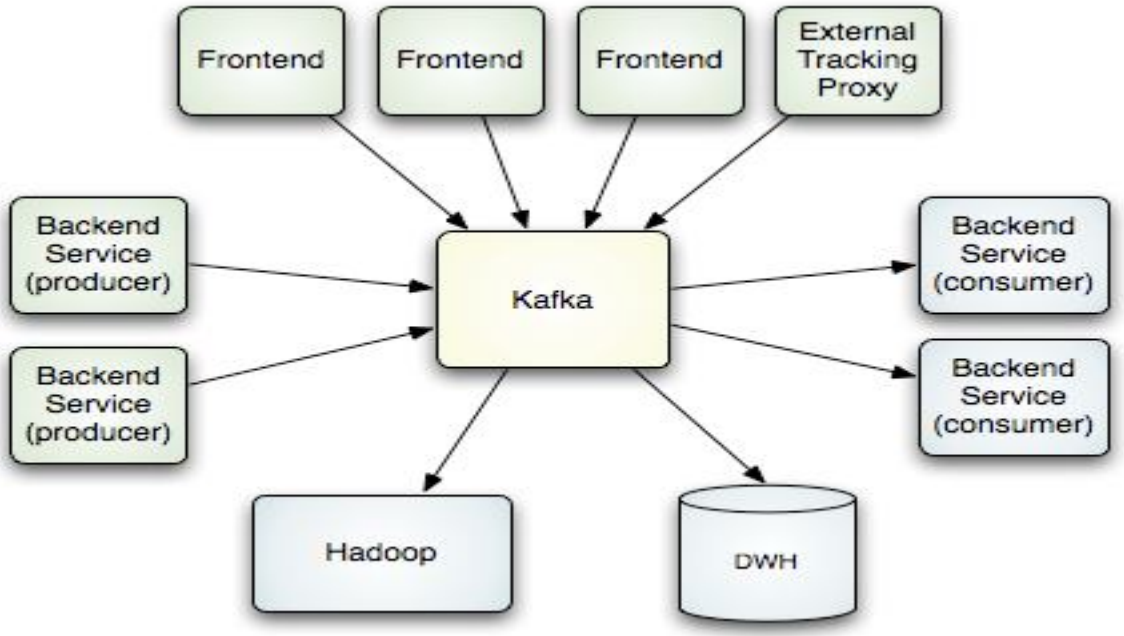
Written in Scala.

Does not follow JMS Standards, neither uses JMS APIs.

Supports both queue and topic semantics.



Overview



Credit : http://kafka.apache.org/design.html

Key terminology

Kafka maintains feeds of messages in categories called ***topic***

Processes that publish messages to a Kafka topic are called ***producers***.

Processes that subscribe to topics and process the feed of published messages are called ***consumers***.

Kafka is run as a cluster comprised of one or more servers each of which is called a ***broker***.

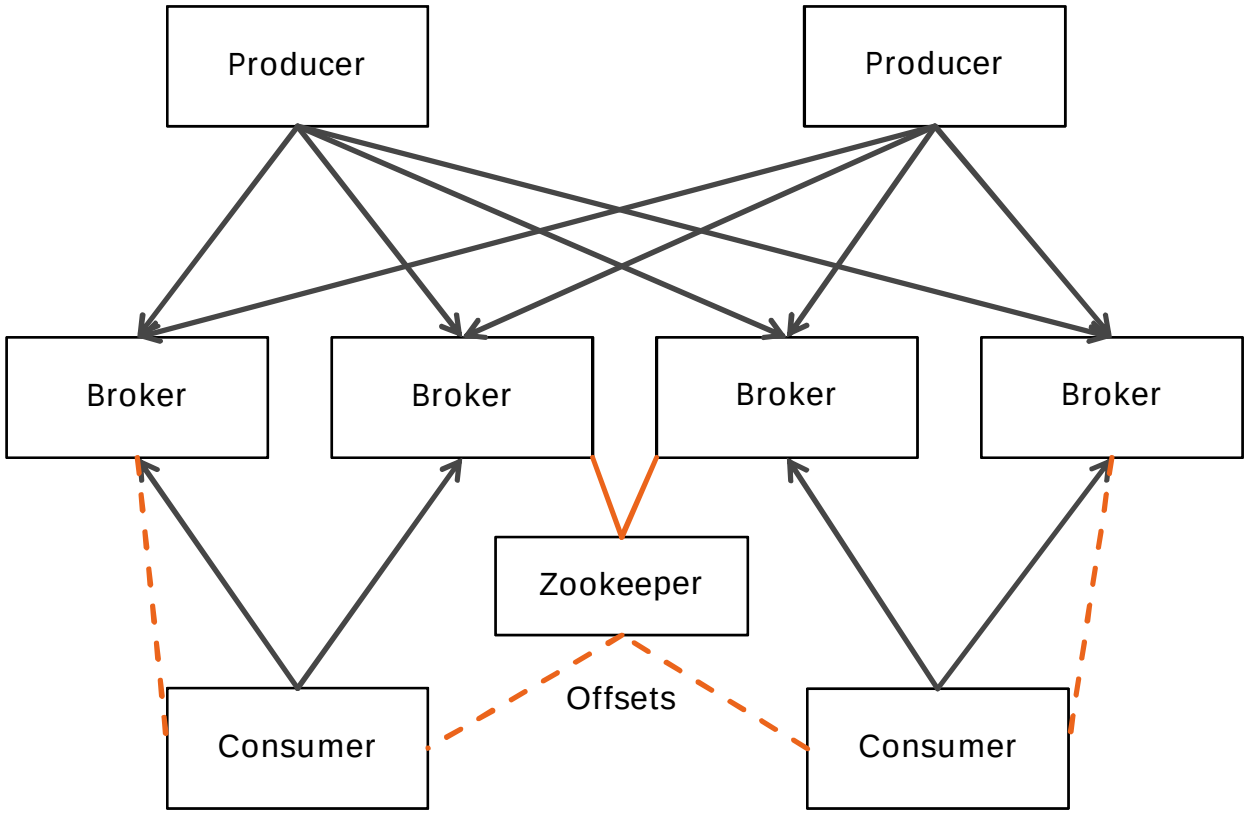
Communication between all components is done via a high performance simple binary API over TCP protocol

Architecture

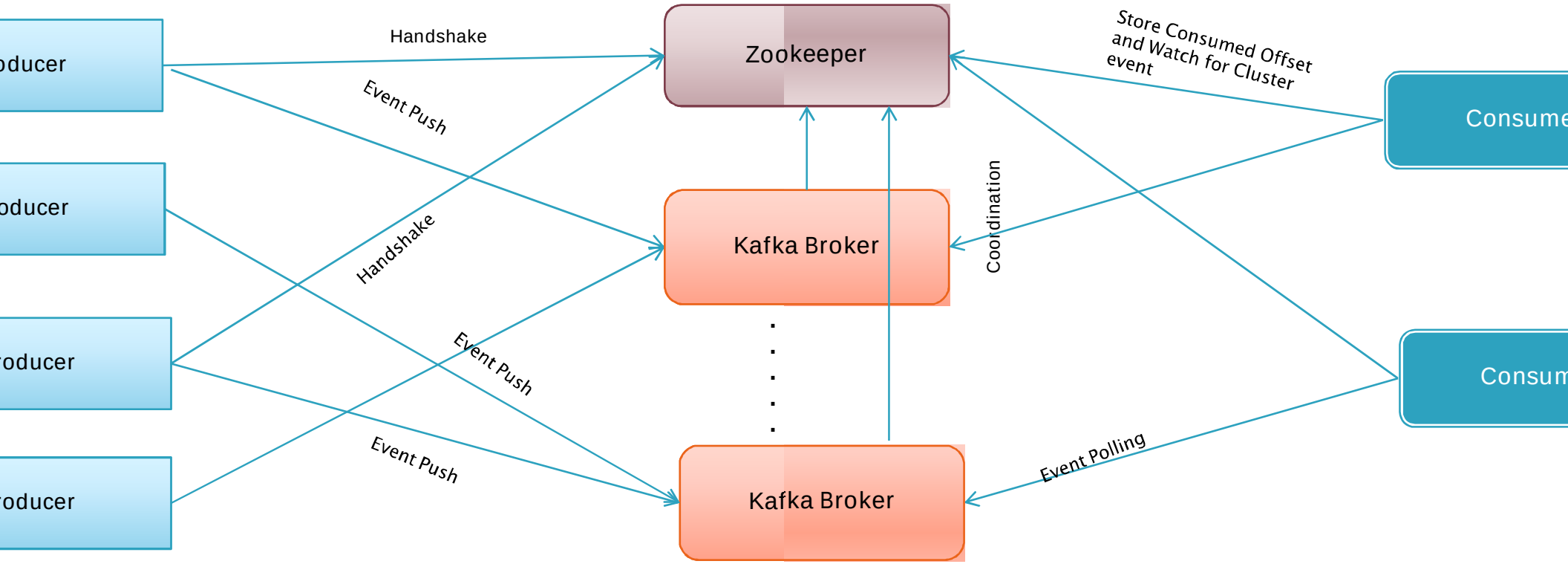
ducers

ka
ster

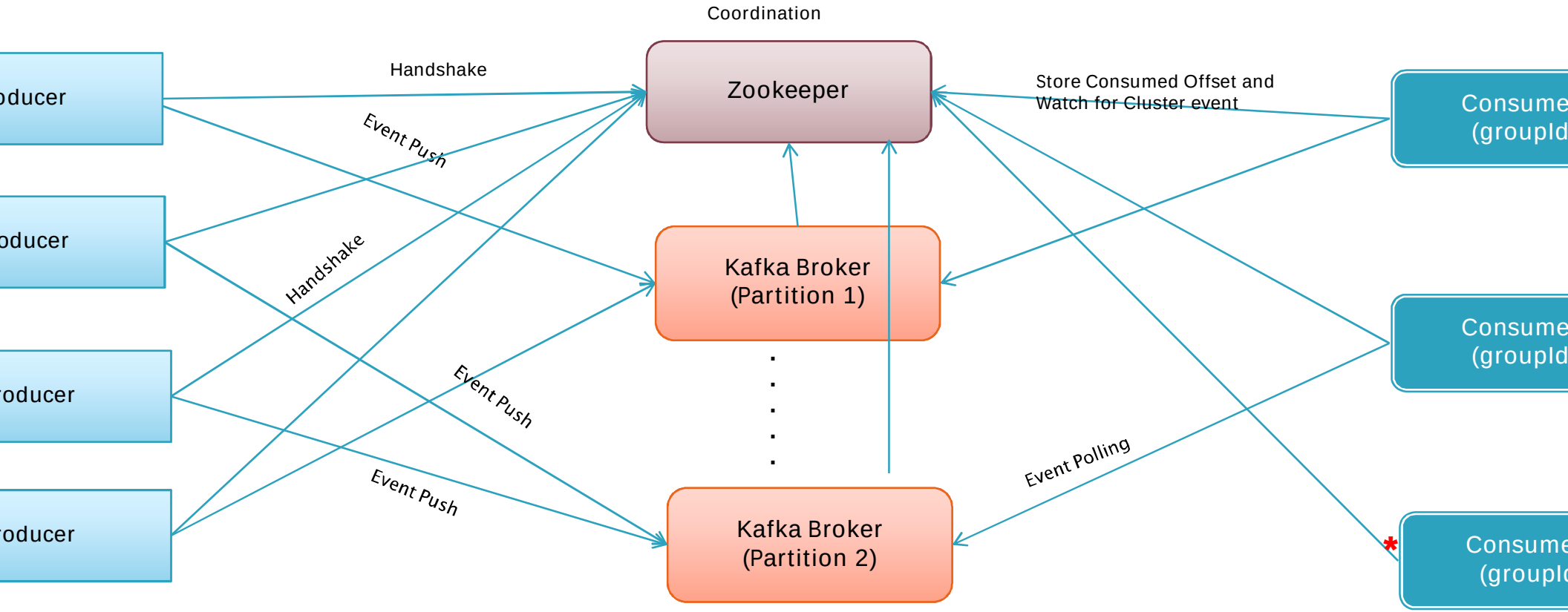
nsumers



How it works


oducer
oducer
roducer
roducer
Handshake
Event Push
Handshake
Event Push
Event Push
Zookeeper
Kafka Broker
Kafka Broker
Coordination
Store Consumed Offset
and Watch for Cluster
event
Event Polling
Consume
Consum

How it works (Queue)



* Consumer 3 would not receive any data, as number of consumers are more than number of partitions.

Design Elements

Filesystem Cache

Zero-copy transfer of messages

Batching of Messages

Batch Compression

Automatic Producer Load balancing.

Broker does not **Push** messages to Consumer, Consumer **Pol** messages from Broker.

And Some others.

- Cluster formation of Broker/Consumer using Zookeeper, So on the fly more consumer, broker can be introduced. The new cluste rebalancing will be taken care by Zookeeper
- Data is persisted in broker and is not removed on consumption (till retention period), so if one consumer fails while consuming, s message can be re-consume again later from broker.
- Simplified storage mechanism for message, not for each message per consumer.

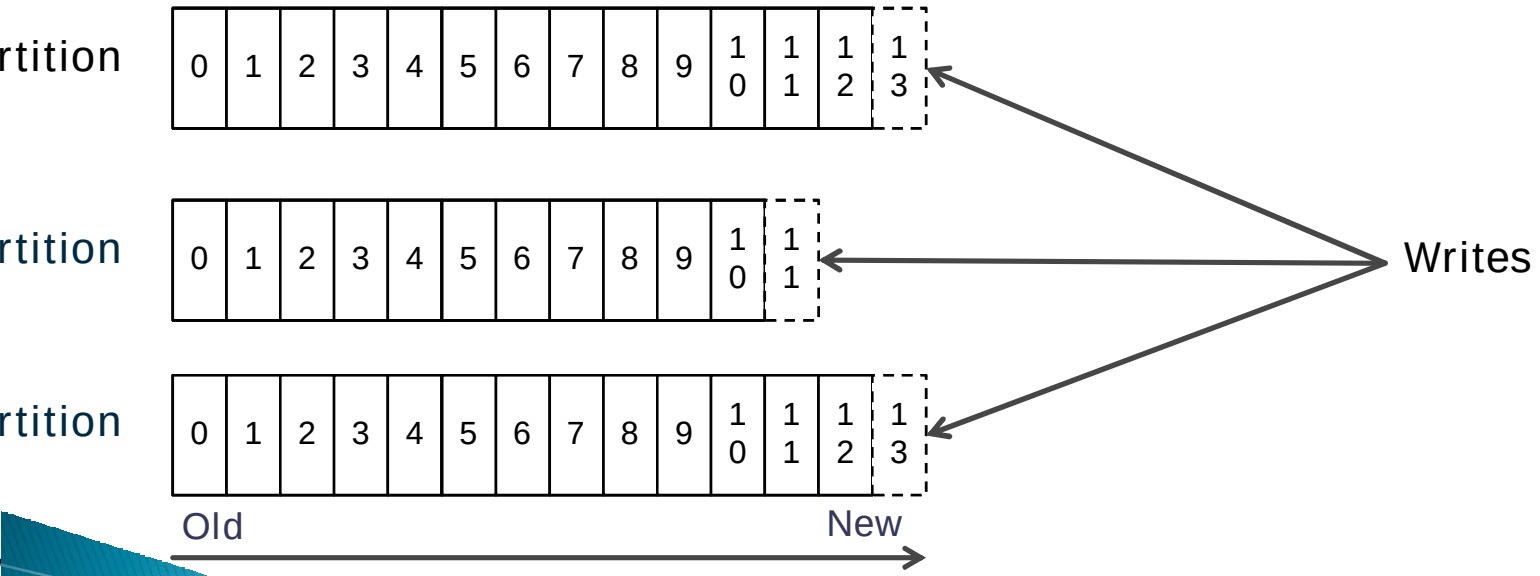
Topics – Partitions

opics are broken up into ordered commit logs called partitions.

ach message in a partition is assigned a sequential id called a

ffset.

ata is retained for a configurable period of time*

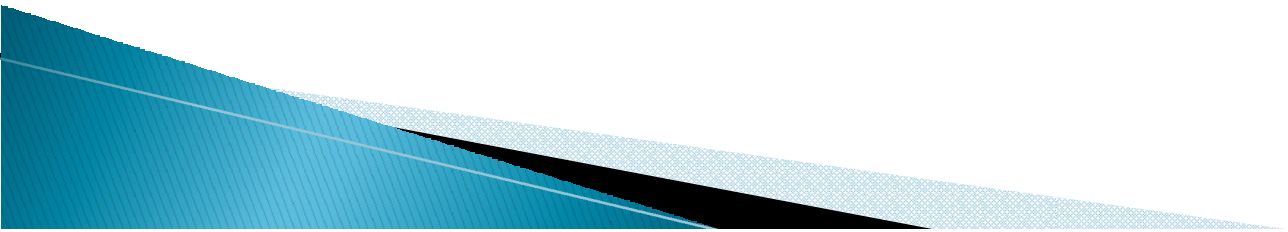


Message Ordering

Ordering is only guaranteed within a partition for a topic

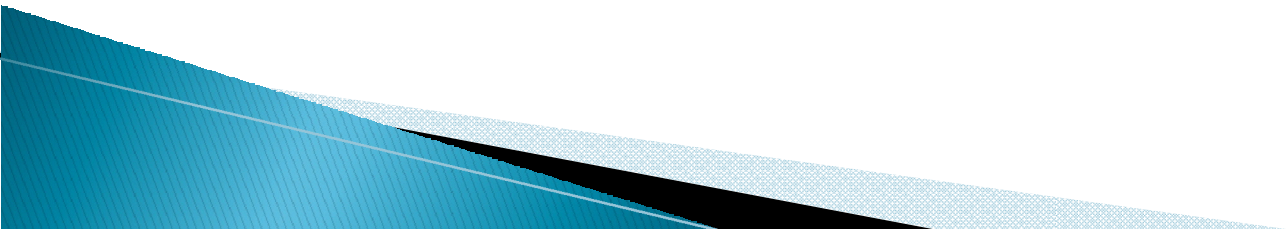
To ensure ordering:

- Group messages in a partition by key (producer)
- Configure exactly one consumer instance per partition within a consumer group



Guarantees

- Messages sent by a producer to a particular topic partition w be appended in the order they are sent
- A consumer instance sees messages in the order they are stored in the log
- For a topic with replication factor N, Kafka can tolerate up to N-1 server failures without “losing” any messages committed to the log



Topics – Replication

Topics can (and should) be replicated.

The unit of replication is the partition

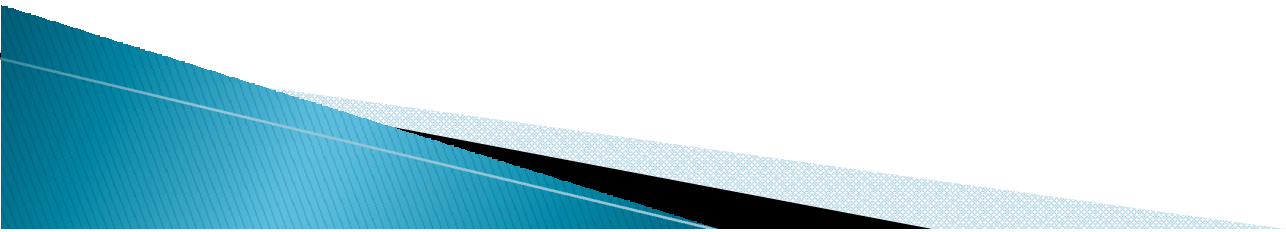
Each partition in a topic has 1 leader and 0 or more replicas.

A replica is deemed to be “in-sync” if

- The replica can communicate with Zookeeper
- The replica is not “too far” behind the leader (configurable)

The group of in-sync replicas for a partition is called the ***ISR*** (In-Sync Replicas)

The Replication factor cannot be lowered

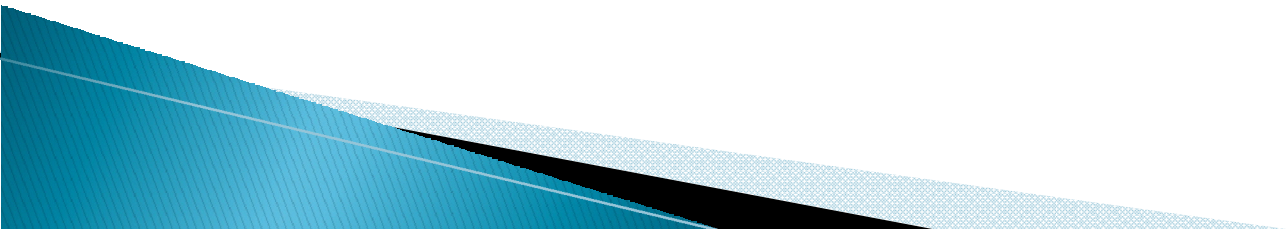


Topics – Replication

urability can be configured with the producer configuration
equest.required.acks

- ***0*** The producer never waits for an ack
- ***1*** The producer gets an ack after the leader replica has received the d
- ***-1*** The producer gets an ack after all ISRs receive the data

Minimum available ISR can also be configured such that an err
s returned if enough replicas are not available to replicate dat



Durable Writes

roducers can choose to *trade* throughput for durability of writes:

rabilit	Behaviour	Per Event Latency	Required Acknowledgements (request.required.acks)
hest	ACK all ISRs have received	Highest	-1
dium	ACK once the leader has received	Medium	1
west	No ACKs required	Lowest	0

Throughput can also be raised with ***more brokers***... (so do this instead)!

A sane configuration:

Property	Value
replication	3
min.insync.replicas	2
request.required.acks	-1

Producer

roducers publish to a topic of their choosing (push)

oad can be distributed

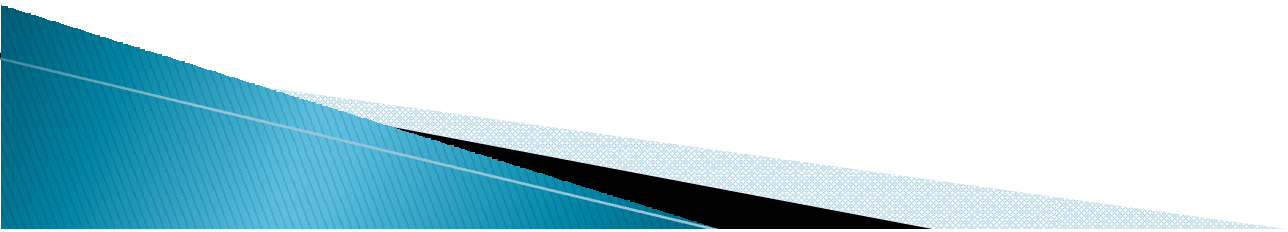
- Typically by “round-robin”
- Can also do “semantic partitioning” based on a key in the message

rokers load balance by partition

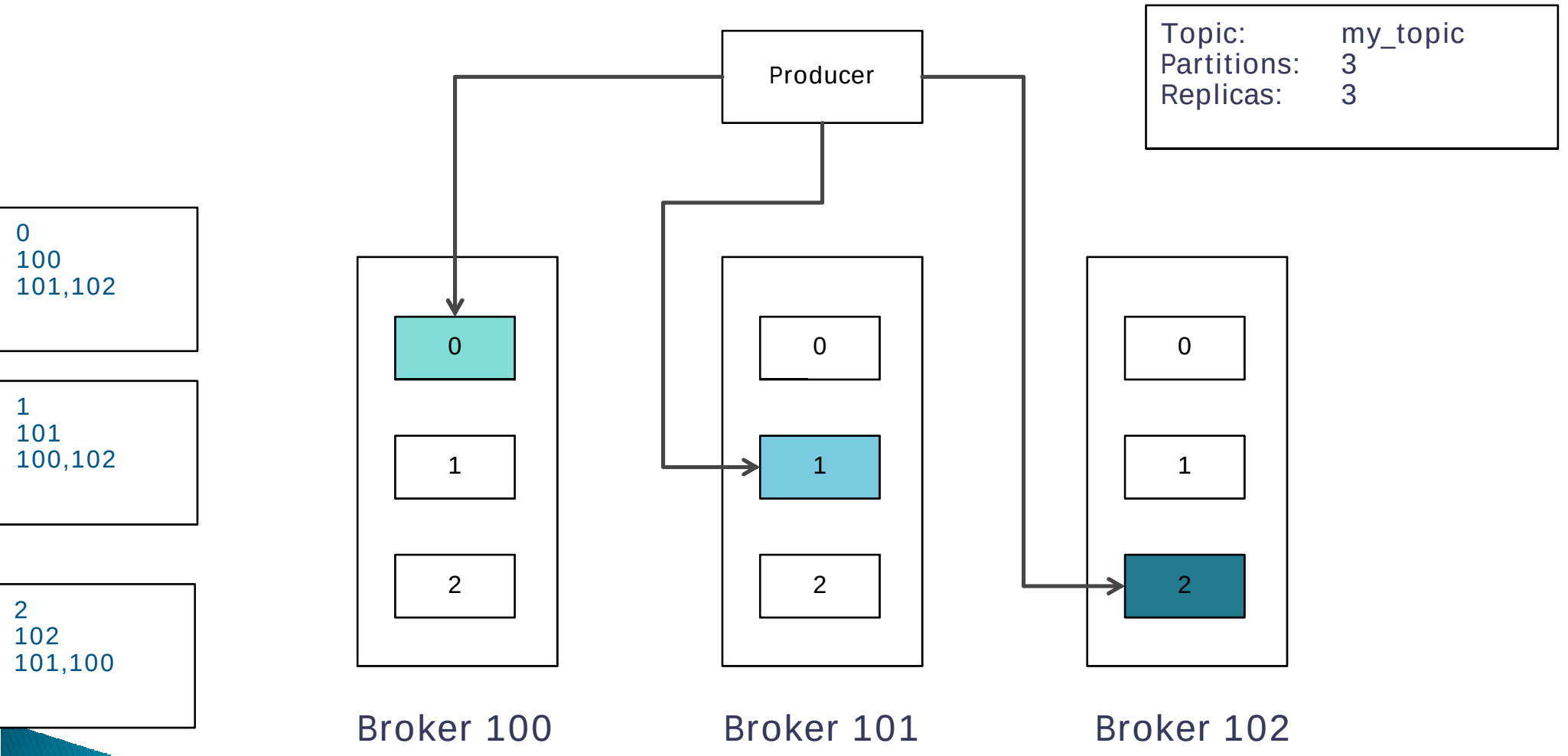
an support async (less durable) sending

ll nodes can answer metadata requests about:

- Which servers are alive
- Where leaders are for the partitions of a topic



Producer – Load Balancing and ISRs



Consumer – Groups

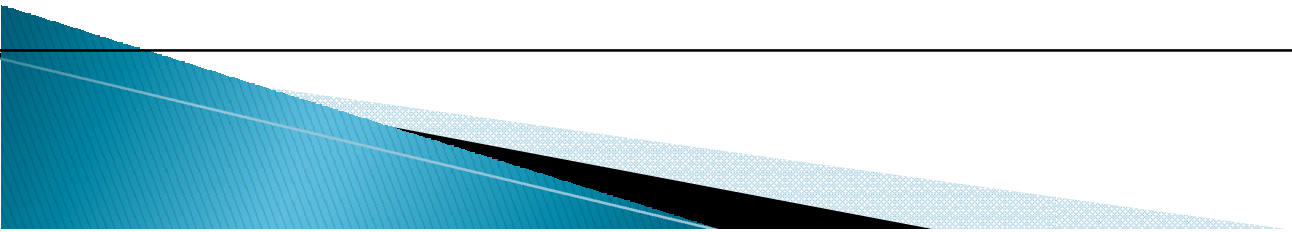
Consumers can be organized into Consumer Groups

ommon Patterns:

1) All consumer instances in one group
   - Acts like a traditional queue with load balancing

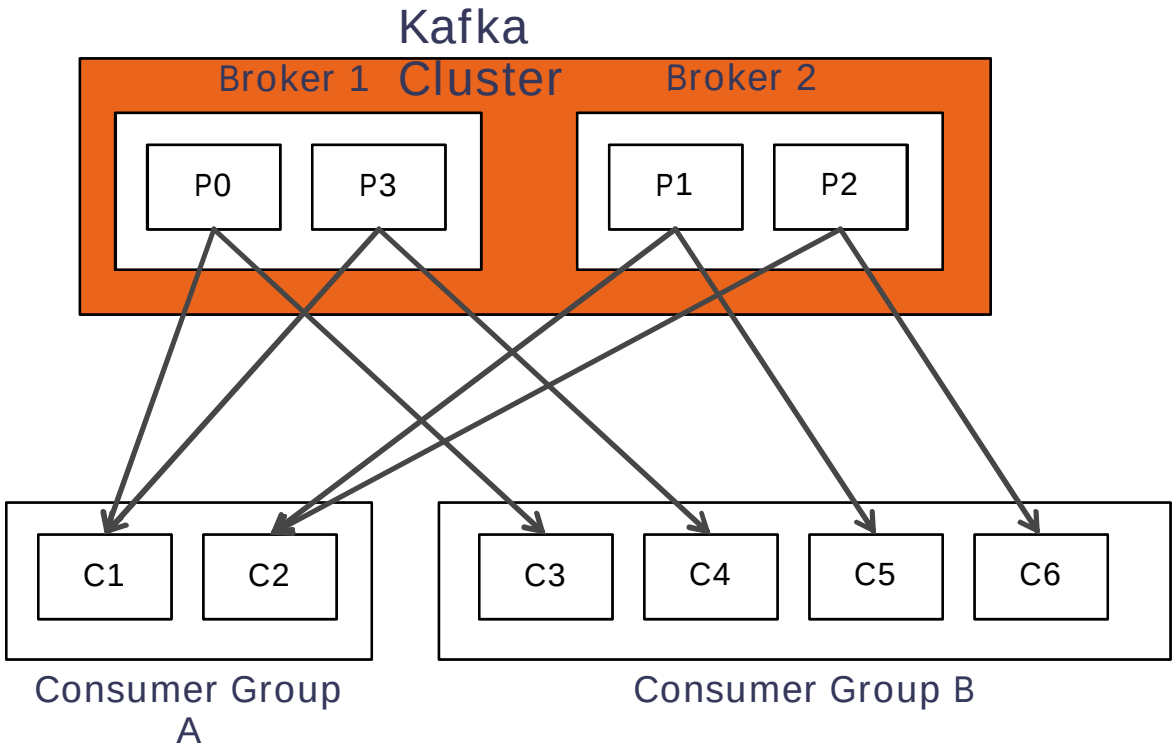
2) All consumer instances in different groups
   - All messages are broadcast to all consumer instances

3) "Logical Subscriber" – Many consumer instances in a grou
   - Consumers are added for scalability and fault tolerance
   - Each consumer instance reads from one or more partitions for a topi
   - There cannot be more consumer instances than partitions



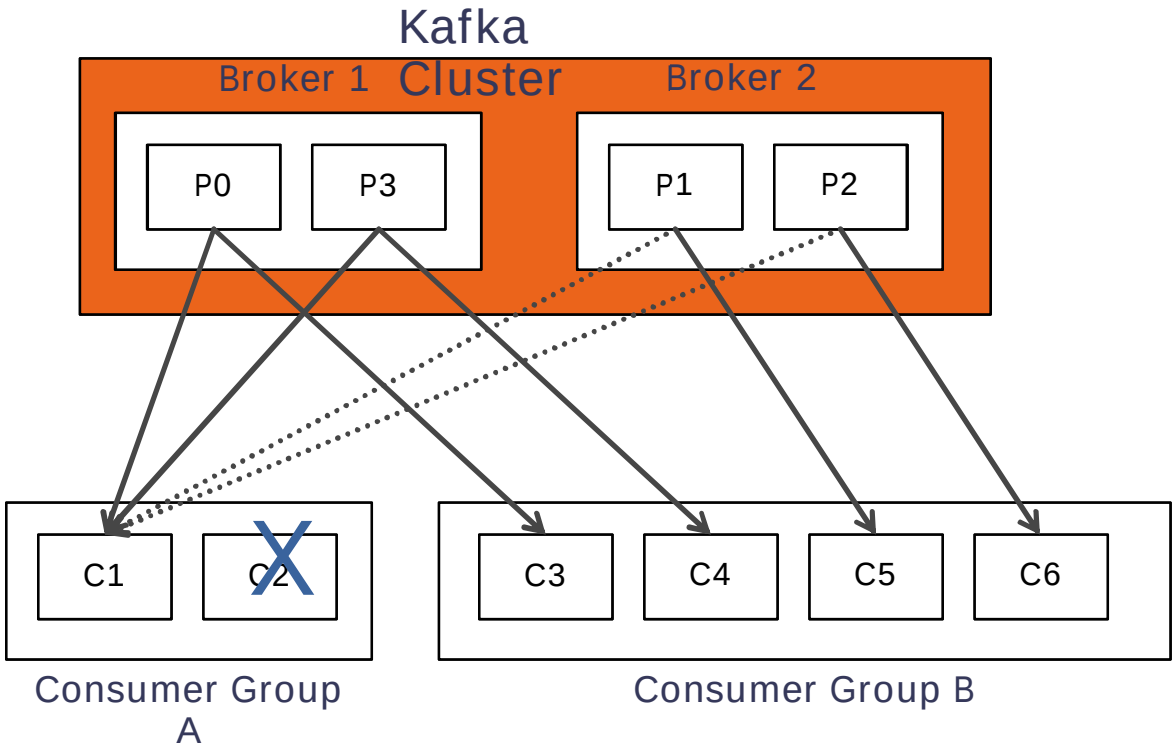
Consumer – Groups

umer
os provide
on to
s and
ons



Consumer – Groups

ebalance
selves



Delivery Semantics

Default

At least once

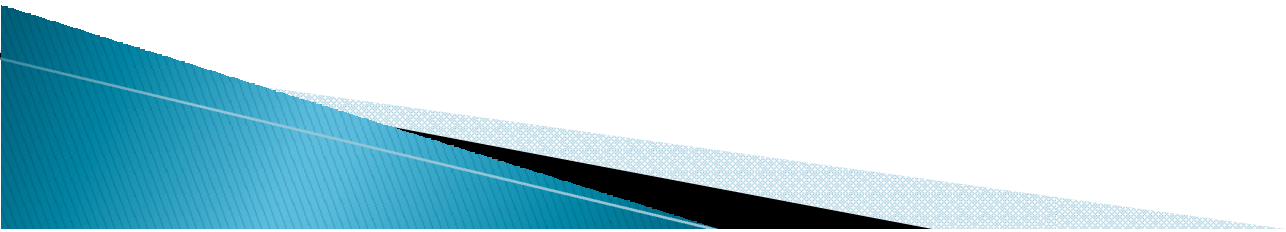
- **Messages are never lost but may be redelivered**

At most once

- Messages are lost but never redelivered

Exactly once

- Messages are delivered once and only once



Delivery Semantics

At least once

- **Messages are never lost but may be redelivered**

At most once

- Messages are lost but never redelivered

Exactly once

- Messages are delivered once and only once

Much Harder
(Impossible??)

Getting Exactly Once Semantics

Must consider two components

- Durability guarantees when *publishing* a message
- Durability guarantees when *consuming* a message

Producer

- What happens when a produce request was sent but a network error returned before an ack?
- Use a single writer per partition and check the latest committed value after network errors

Consumer

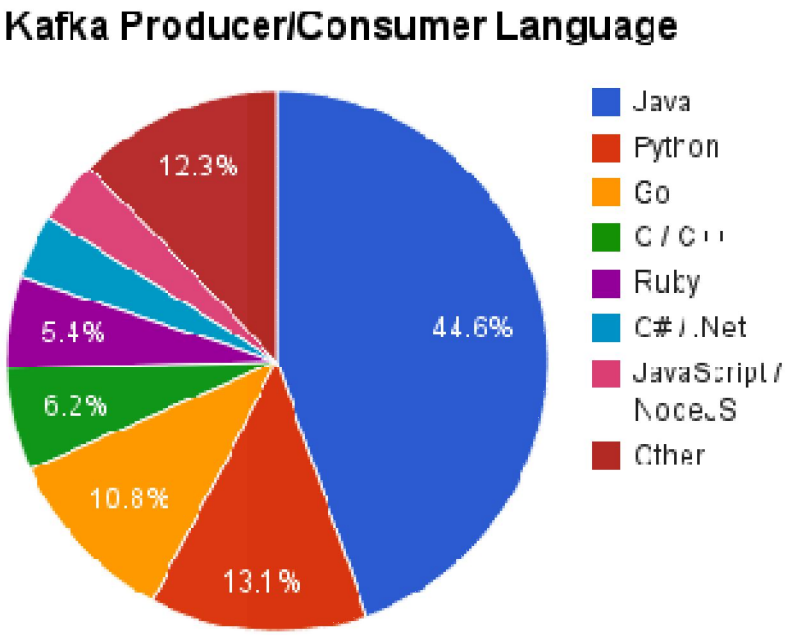
- Include a unique ID (e.g. UUID) and de-duplicate.
- Consider storing offsets with data

Kafka Clients

Remember Kafka implements a binary TCP protocol.

All clients except the JVM-based clients are maintained external to the code base.

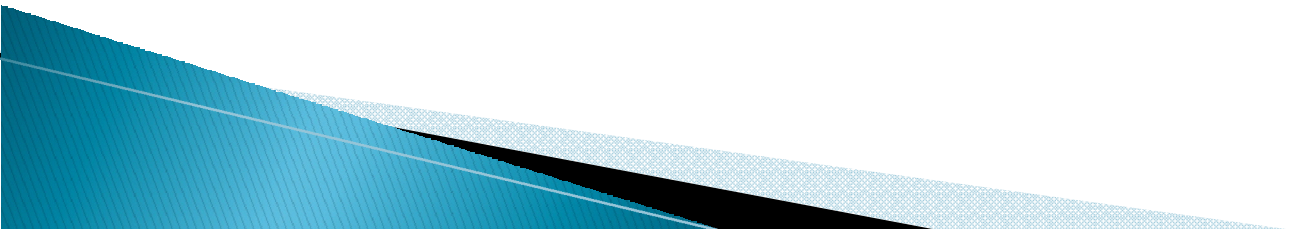
Full Client List he



Java Producer Example – Old (< 0.8.1)

```
  /* start the producer */
private void start() {
producer = new Producer<String, String]]>(config);
}

/* create record and send to Kafka
* because the key is null, data will be sent to a random partition.
* the producer will switch to a different random partition every 10
tes
**/
private void produce(String s) {
KeyedMessage<String, String]]> message = new
dMessage<String, String]]>(topic, null, s);
producer.send(message);
}
```



Producer – New

Send the given record asynchronously and return a future which will
ually contain the response information.

@param record The record to send
@return A future which will eventually contain the response information

blic Future send(ProducerRecord record);

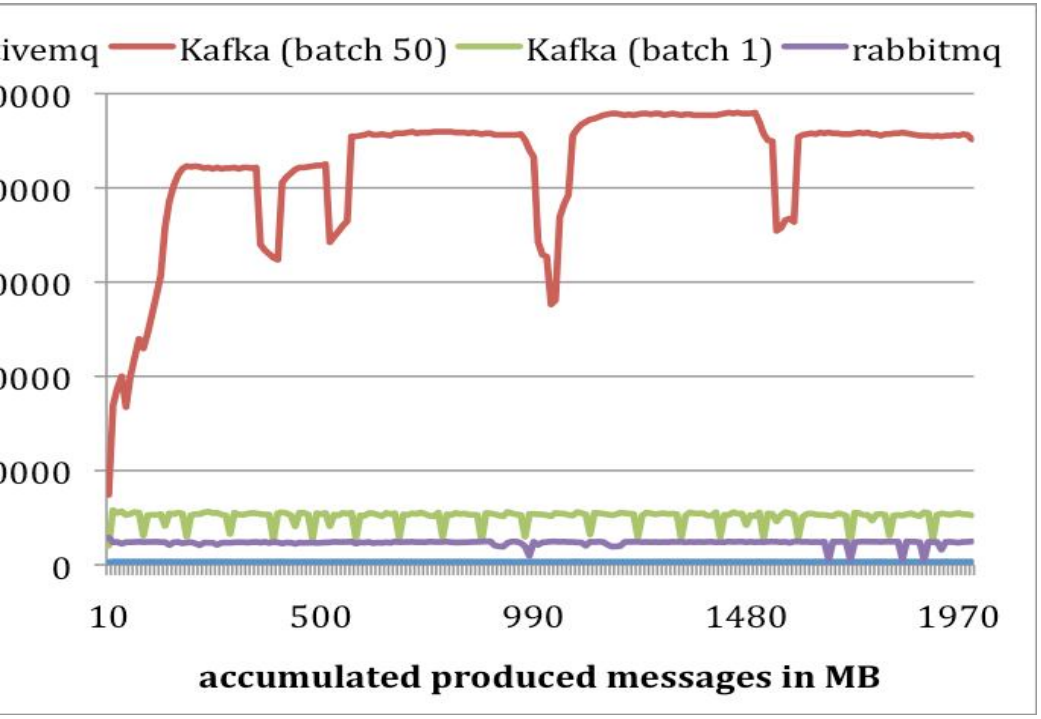
Send a record and invoke the given callback when the record has been
owledged by the server

blic Future send(ProducerRecord record, Callback callback);

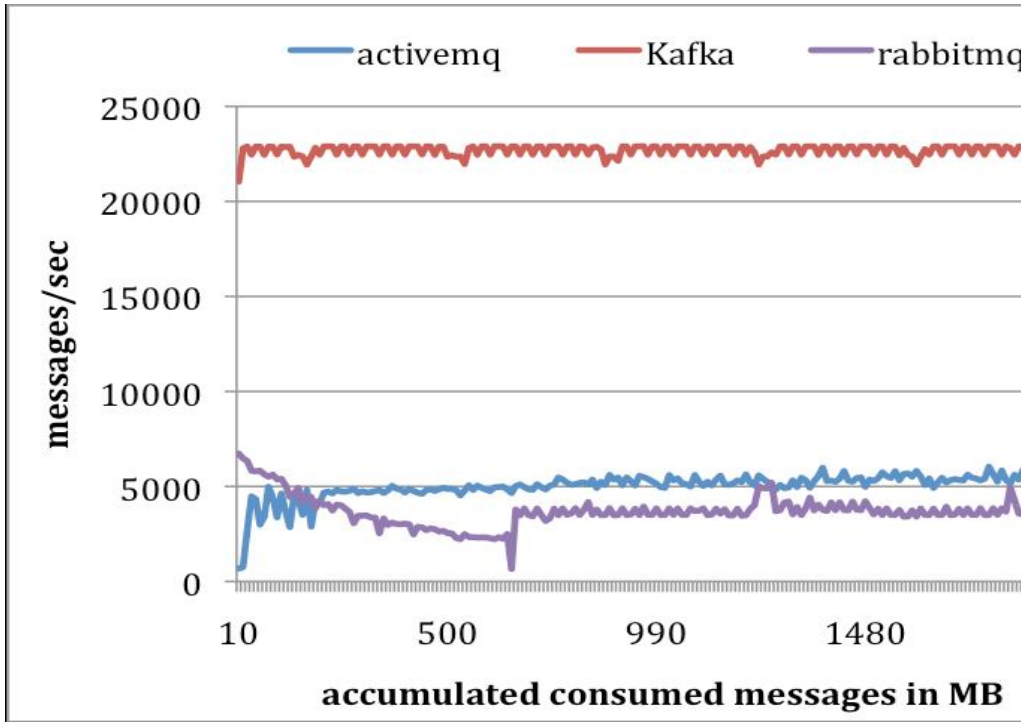
```
/ configure
roperties config = new Properties();
onfig.setProperty(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
afkaProducer producer = new KafkaProducer(config);

/ create and send a record
roducerRecord record = new ProducerRecord("topic", "key".getBytes(), "value".getBytes());
uture<RecordMetadata> response = producer.send(record); // this is always non-blocking
ystem.out.println("The offset was: " + response.get().offset()); // get() blocks
```

Performance Numbers



Producer Performance



Consumer Performance

Credit : http://research.microsoft.com/en-us/UM/people/srikanth/netdb11/netdb11papers/netdb11-final12.pdf

Powered By

ache Kafka is used at LinkedIn for activity stream data and operational metrics. This powers various products like LinkedIn Newsfeed, LinkedIn Today in addition to our offline analytics systems like Hadoop.
this
c. - Apache kafka is used at Mate1 as our main event bus that powers our news and activity feeds, automated review systems, and will soon power real time notifications and log distribution.
che Kafka drives our new pub sub system which delivers real-time events for users in our latest game - Deckadence. It will soon be used in a host of new use cases including group chat and back end stats and log collection.
pache Kafka aggregates high-flow message streams into a unified distributed pubsub service, brokering the data for other internal systems as part of Boundary's real-time network analytics infrastructure.
ache Kafka is used at DataSift as a collector of monitoring events and to track user's consumption of data streams in real time. http://highscalability.com/blog/2011/11/29/datasift-architecture-realtime-datamining-at-120000-tweets
use Kafka to aggregate and process tracking data from all our facebook games (which are hosted at various providers) in a central location.
ache Kafka is used at AddThis to collect events generated by our data network and broker that data to our analytics clusters and real-time web analytics platform.
- At Urban Airship we use Kafka to buffer incoming data points from mobile devices for processing by our analytics infrastructure.
- We use Kafka to collect realtime event data from clients, as well as our own internal service metrics, that feed our interactive analytics dashboards.
We use Kafka internally as part of our reliable email queueing system.
We use a hierarchical distributed counting engine, uses Kafka as a primary speedy interface as well as routing events for cascading counting
We use Kafka to collect all metrics and events generated by the users of the website.
art of their Storm stream processing infrastructure, e.g. this.
this blog.
Kafka is part of the InfoChimps real-time data platform.
this blog.
ue - We use Kafka as a distributed queue in front of our web traffic stream processing infrastructure (Storm).
is used as the primary high speed message queue to power Storm and our real-time analytics/event ingestion pipelines.
Kafka powers online to online messaging, and online to offline messaging at Foursquare. We integrate with monitoring, production systems, and our offline infrastructure, including hadoop.
ka brokers data to most systems in our metrics and events ingestion pipeline. Different modules contribute and consume data from it, for streaming CEP (homegrown), persistence (at different "temperatures" in Redis, ElasticSea
ysis (Hadoop).
e use Kafka 1. as an infrastructure that helps us bring continuously the tracking events from various datacenters into our central hadoop cluster for offline processing, 2. as a propagation path for data integration, 3. as a real-time
recommendation engines
SPM (performance monitoring), Kafka is used for performance metrics collection and feeds SPM's in-memory data aggregation (OLAP cube creation) as well as our CEP/Alerts servers. In SA (search analytics) Kafka is used in s
ore being aggregated and persisted in HBase.
a will soon be replacing part of our current production system to collect performance and usage data from the end-users browser for projects like Telemetry, Test Pilot, etc. Downstream consumers usually persist to either HDFS
rce - At Wize Commerce (previously, NexTag), Kafka is used as a distributed queue in front of Storm based processing for search index generation. We plan to also use it for collecting user generated data on our web tier, landing
e Hadoop, HBase, etc.
uixey, The Search Engine for Apps, Kafka is an integral part of our eventing, logging and messaging infrastructure.
afka is used at LinkSmart as an event stream feeding Hadoop and custom real time systems.
use Kafka at Simple for log aggregation and to power our analytics infrastructure.
ig Data - We use Kafka for syncing LucidWorks Search (Solr) with incoming data from Hadoop and also for sending LucidWorks Search logs back to Hadoop for analysis.
se Kafka as a bus to move all systems events through our various datacenters. This includes metrics, logs, custom events etc. On the consumer side, we output into Splunk, Graphite, Esper-like real-time alerting.

Credit: https://cwiki.apache.org/confluence/display/KAFKA/Powered+By