KAFKA ARCHITECTURE

May 11, 2017

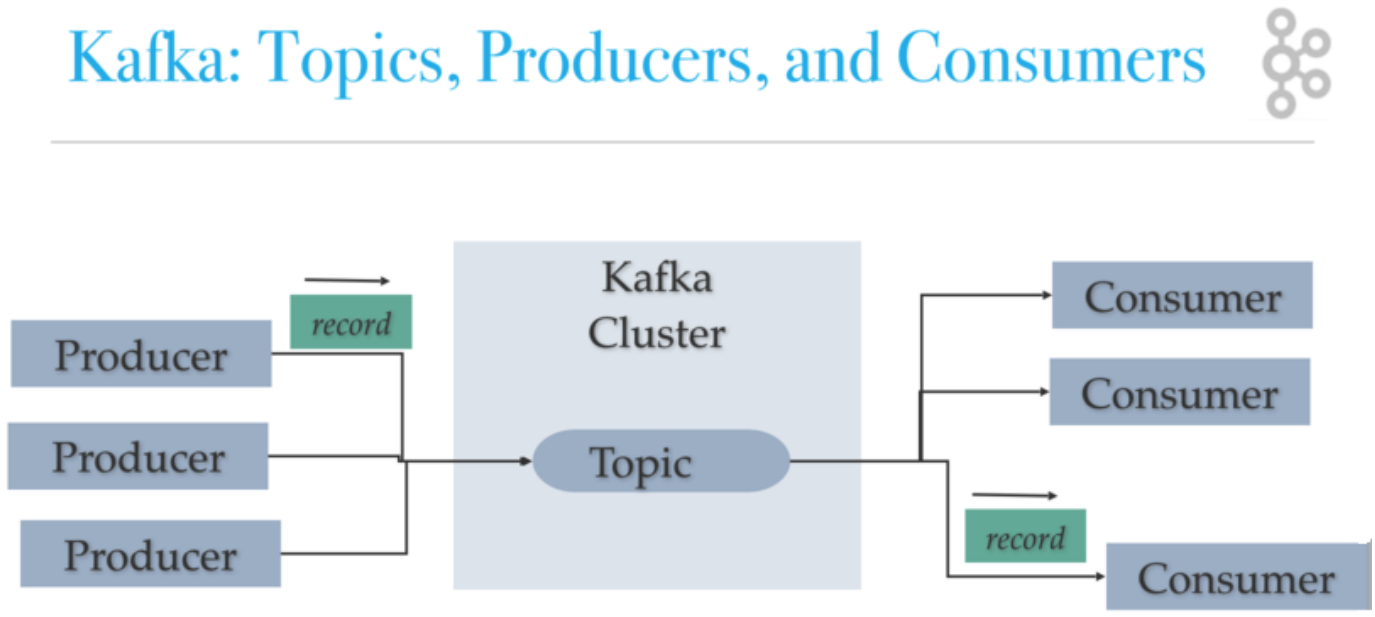
If you are not sure what Kafka is, see What is Kafka? (http://cloudurable.com/blog/what-is-kafka/index.html).

Kafka Architecture

Kafka (http://cloudurable.com/kafka-training/index.html) consists of Records, Topics, Consumers, Producers, Brokers, Logs, Partitions, and Clusters. Records can have key (optional), value and timestamp. Kafka Records are immutable. A Kafka Topic is a stream of records (`"/orders"` , `"/user-signups"`). You can think of a Topic as a feed name. A topic has a Log which is the topic's storage on disk. A Topic Log is broken up into partitions and segments. The Kafka Producer API is used to produce streams of data records. The Kafka Consumer API is used to consume a stream of records from Kafka. A Broker is a Kafka server that runs in a Kafka Cluster. Kafka Brokers form a cluster. The Kafka Cluster consists of many Kafka Brokers on many servers. Broker sometimes refer to more of a logical system or as Kafka as a whole.

> Cloudurable provides Kafka training (http://cloudurable.com/kafka-training/index.html), Kafka consulting (http://cloudurable.com/kafka-aws-consulting/index.html), Kafka support (http://cloudurable.com/subscription_support/index.html) and helps setting up Kafka clusters in AWS (http://cloudurable.com/services/index.html).

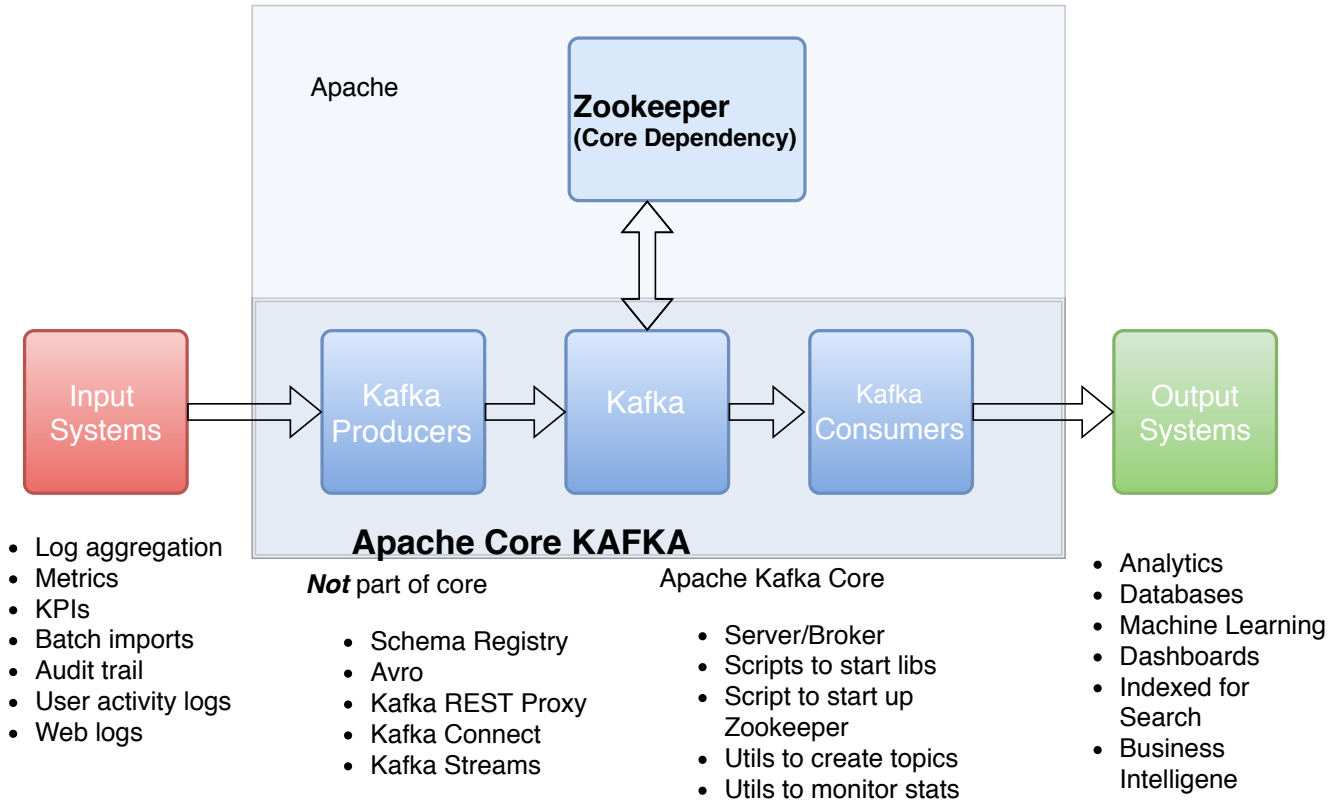
Kafka Architecture: Topics, Producers and Consumers



"

Kafka (http://cloudurable.com/kafka-aws-consulting/index.html) uses ZooKeeper to manage the cluster. ZooKeeper is used to coordinate the brokers/cluster topology. ZooKeeper is a consistent file system for configuration information. ZooKeeper gets used for leadership election for Broker Topic Partition Leaders.

Kafka Architecture: Core Kafka



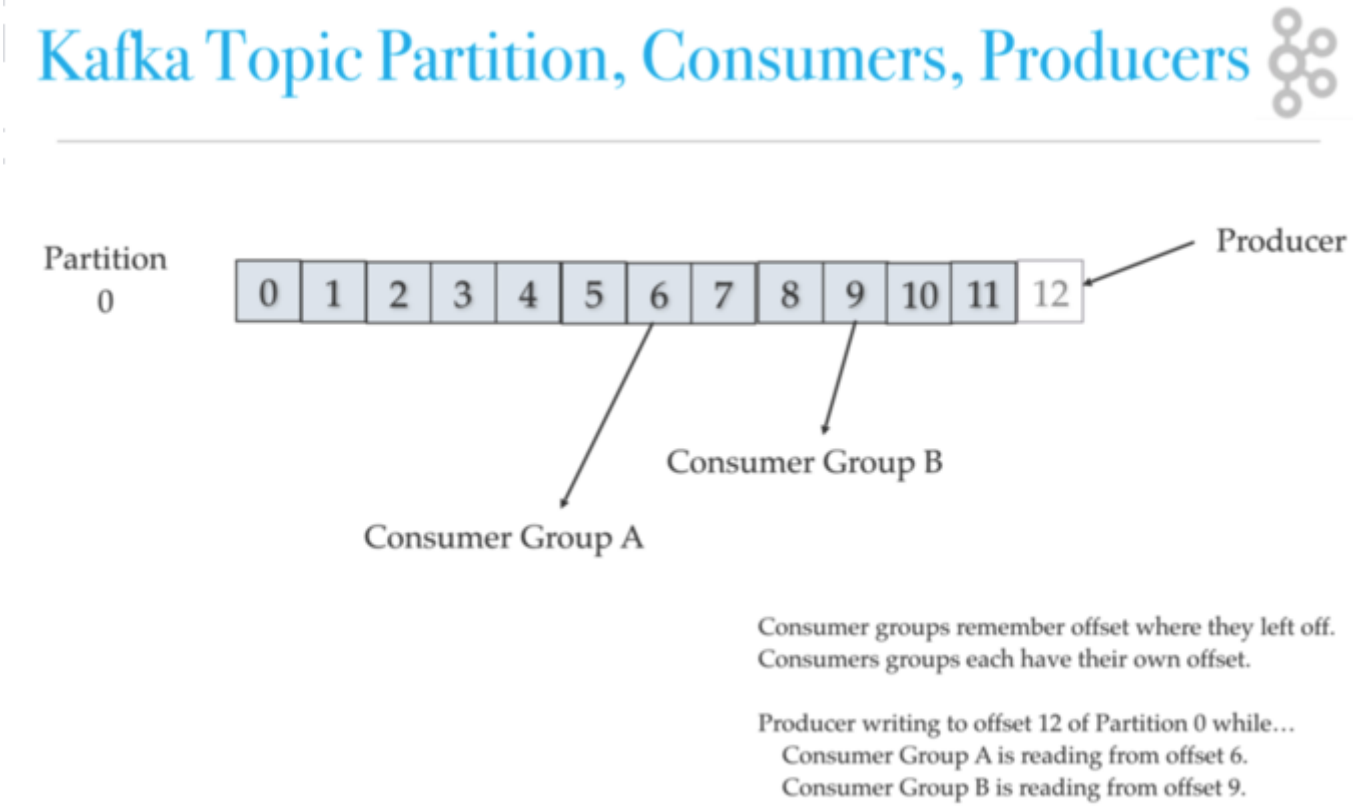
Kafka needs ZooKeeper

Kafka uses Zookeeper to do leadership election of Kafka Broker and Topic Partition pairs. Kafka uses Zookeeper to manage service discovery for Kafka Brokers that form the cluster. Zookeeper sends changes of the topology to Kafka, so each node in the cluster knows when a new broker joined, a Broker died, a topic was removed or a topic was added, etc. Zookeeper provides an in-sync view of Kafka Cluster configuration.

Kafka Producer, Consumer, Topic details

Kafka producers write to Topics. Kafka consumers read from Topics. A topic is associated with a log which is data structure on disk. Kafka appends records from a producer(s) to the end of a topic log. A topic log consists of many partitions that are spread over multiple files which can be spread on multiple Kafka cluster nodes. Consumers read from Kafka topics at their cadence and can pick where they are (offset) in the topic log. Each consumer group tracks offset from where they left off reading. Kafka distributes topic log partitions on different nodes in a cluster for high performance with horizontal scalability. Spreading partitions aids in writing data quickly. Topic log partitions are Kafka way to shard reads and writes to the topic log. Also, partitions are needed to have multiple consumers in a consumer group work at the same time. Kafka replicates partitions to many nodes to provide failover.

Kafka Architecture: Topic Partition, Consumer group, Offset and Producers



Kafka Scale and Speed

How can Kafka scale if multiple *producers* and *consumers* read and write to same Kafka topic log at the same time? First Kafka is fast, Kafka writes to filesystem sequentially which is fast. On a modern fast drive, Kafka can easily write up to 700 MB or more bytes of data a second. Kafka scales writes and reads by *sharding topic logs into partitions*. Recall topics logs can be split into multiple partitions which can be stored on multiple different servers, and those servers can use multiple disks. Multiple producers can write to different *partitions* of the same topic. Multiple consumers from multiple *consumer groups* can read from different partitions efficiently.

Kafka Brokers

A *Kafka cluster* is made up of multiple Kafka Brokers. Each Kafka Broker has a unique ID (number). Kafka Brokers contain topic log partitions. Connecting to one broker bootstraps a client to the entire Kafka cluster. For failover, you want to start with at least three to five brokers. A Kafka cluster can have, 10, 100, or 1,000 brokers in a cluster if needed.

Kafka Cluster, Failover, ISRs

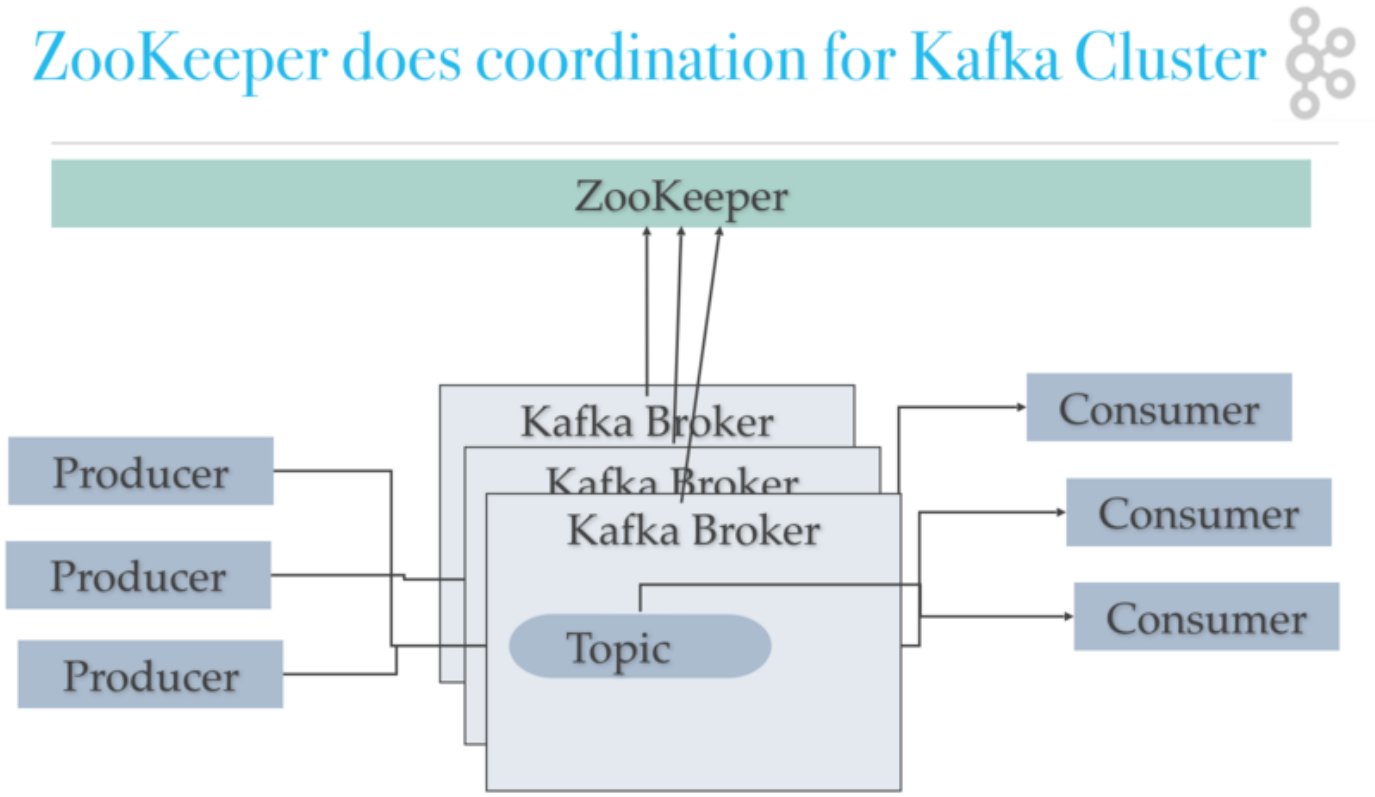
Kafka supports replication to support failover. Recall that Kafka uses ZooKeeper to form Kafka Brokers into a cluster and each node in Kafka cluster is called a Kafka Broker. Topic partitions can be replicated across multiple nodes for failover. The topic should have a replication factor greater than 1 (2, or 3). For example, if you are running in AWS, you would want to be able to survive a single availability zone outage. If one Kafka Broker goes down, then the Kafka Broker which is an ISR (in-sync replica) can serve data.

Kafka Failover vs. Kafka Disaster Recovery

Kafka uses replication for failover. Replication of Kafka topic log partitions allows for failure of a rack or AWS availability zone (AZ). You need a replication factor of at least 3 to survive a single AZ failure. You need to use Mirror Maker, a Kafka utility that ships with Kafka core, for disaster recovery. Mirror Maker replicates a Kafka cluster to another data-center or AWS region. They call what Mirror Maker does mirroring as not to be confused with replication.

Note there is no hard and fast rule on how you have to set up the Kafka cluster per se. You could, for example, set up the whole cluster in a single AZ so you can use AWS enhanced networking and placement groups for higher throughput, and then use Mirror Maker to mirror the cluster to another AZ in the same region as a hot-standby.

Kafka Architecture: Kafka Zookeeper Coordination



Kafka Topics Architecture

Please continue reading about Kafka Architecture. The next article covers Kafka Topics Architecture (http://cloudurable.com/blog/kafka-architecture-topics/index.html) with a discussion of how partitions are used for fail-over and parallel processing.

Related content

- What is Kafka? (http://cloudurable.com/blog/what-is-kafka/index.html)
- Kafka Architecture (http://cloudurable.com/blog/kafka-architecture/index.html)
- Kafka Topic Architecture (http://cloudurable.com/blog/kafka-architecture-topics/index.html)
- Kafka Consumer Architecture (http://cloudurable.com/blog/kafka-architecture-consumers/index.html)
- Kafka Producer Architecture (http://cloudurable.com/blog/kafka-architecture-producers/index.html)
- Kafka Architecture and low level design (http://cloudurable.com/blog/kafka-architecture-low-level/index.html)

- Kafka and Schema Registry (http://cloudurable.com/blog/kafka-avro-schema-registry/index.html)
- Kafka and Avro (http://cloudurable.com/blog/avro/index.html)
- Kafka Ecosystem (http://cloudurable.com/blog/kafka-ecosystem/index.html)
- Kafka vs. JMS (http://cloudurable.com/blog/kafka-vs-jms/index.html)
- Kafka versus Kinesis (http://cloudurable.com/blog/kinesis-vs-kafka/index.html)
- Kafka Tutorial: Using Kafka from the command line (http://cloudurable.com/blog/kafka-tutorial-kafka-from-command-line/index.html)
- Kafka Tutorial: Kafka Broker Failover and Consumer Failover (http://cloudurable.com/blog/kafka-tutorial-kafka-failover-kafka-cluster/index.html)
- Kafka Tutorial (http://cloudurable.com/ppt/kafka-tutorial-cloudruable-v2.pdf)
- Kafka Tutorial: Writing a Kafka Producer example in Java (http://cloudurable.com/blog/kafka-tutorial-kafka-producer/index.html)
- Kafka Tutorial: Writing a Kafka Consumer example in Java (http://cloudurable.com/blog/kafka-tutorial-kafka-consumer/index.html)
- Kafka Architecture: Log Compaction (http://cloudurable.com/blog/kafka-architecture-log-compaction/index.html)
- Kafka Architecture: Low-Level PDF Slides (http://cloudurable.com/ppt/4-kafka-detailed-architecture.pdf)

About Cloudurable

We hope you enjoyed this article. Please provide feedback (http://cloudurable.com/contact/index.html). Cloudurable provides Kafka training (http://cloudurable.com/kafka-training/index.html), Kafka consulting (http://cloudurable.com/kafka-aws-consulting/index.html), Kafka support (http://cloudurable.com/subscription_support/index.html) and helps setting up Kafka clusters in AWS (http://cloudurable.com/services/index.html).

Check out our new GoLang course. We provide onsite Go Lang training which is instructor led (http://cloudurable.com/golang-onsite-instructor-led-training/index.html).