Kafka Architecture

Cluster components

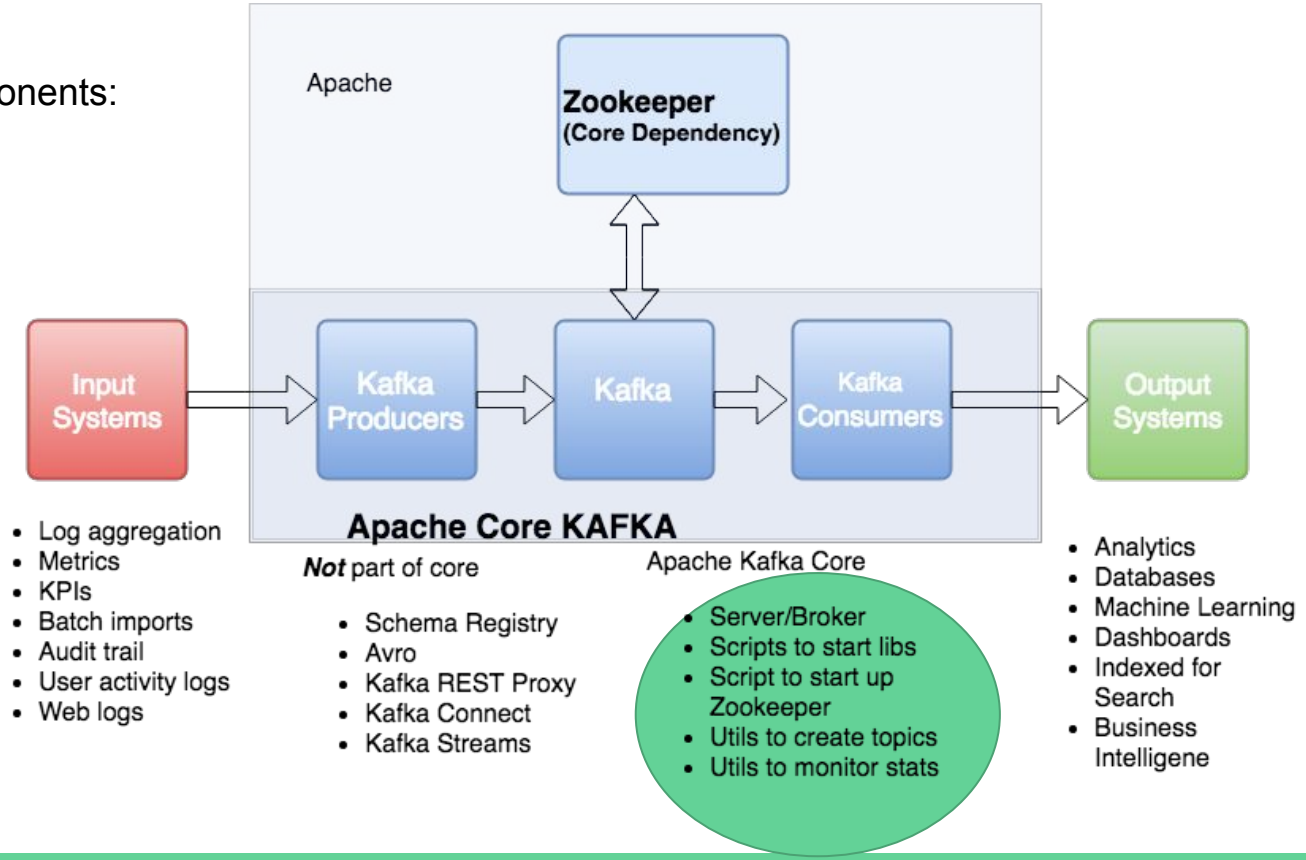
Kafka Architecture

Apache Kafka basic components:
(vanilla Kafka installation)

- Zookeeper
- Brokers
- Producer API
- Consumer API

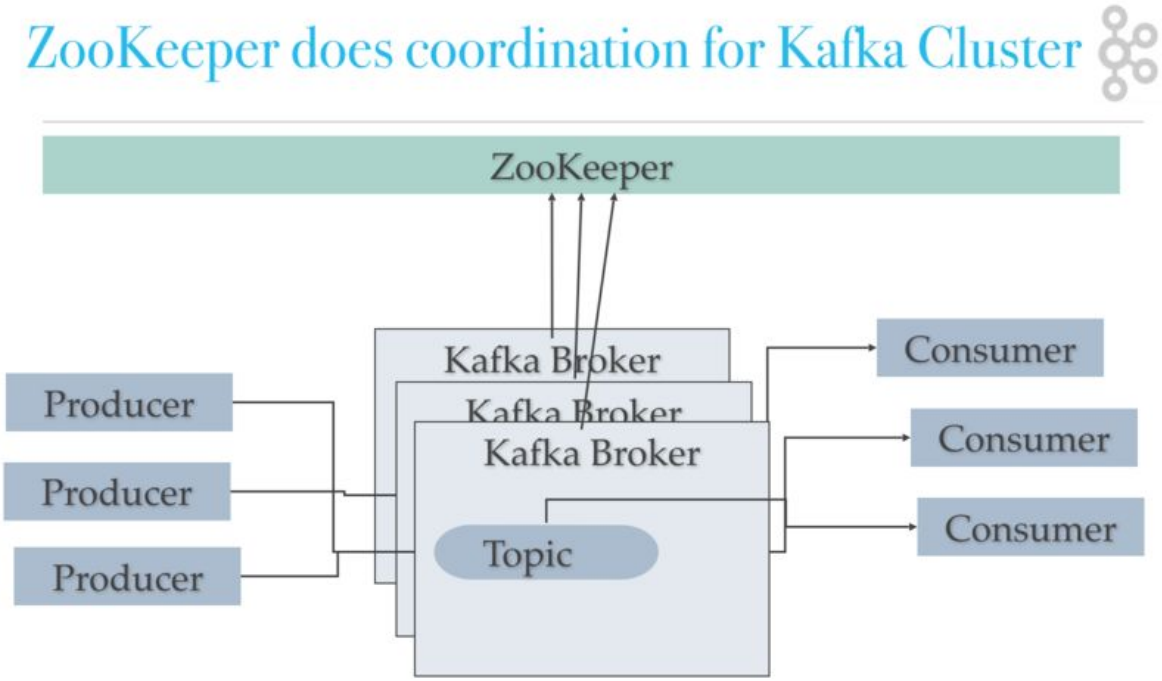
Others:

- REST Proxy
- Schema Registry
- Kafka Connect
- Kafka Streams
- Kafka KSQL



Role of brokers in Kafka Architecture

- Store data
- Active Controller
- Group Coordinator
- Seed/Bootstrap



Apache Zookeeper

coordination service for distributed systems

Zookeeper

Zookeeper is a high-performance coordination service used by distributed systems. Zookeeper allows distributed systems to coordinate each other through a hierarchical name-space of data registrars.

Can be used for:

1. Configuration Management

2. Synchronization

3. Leader Election

4. Notification System

Zookeeper as a distributed (small - 1MB) files system

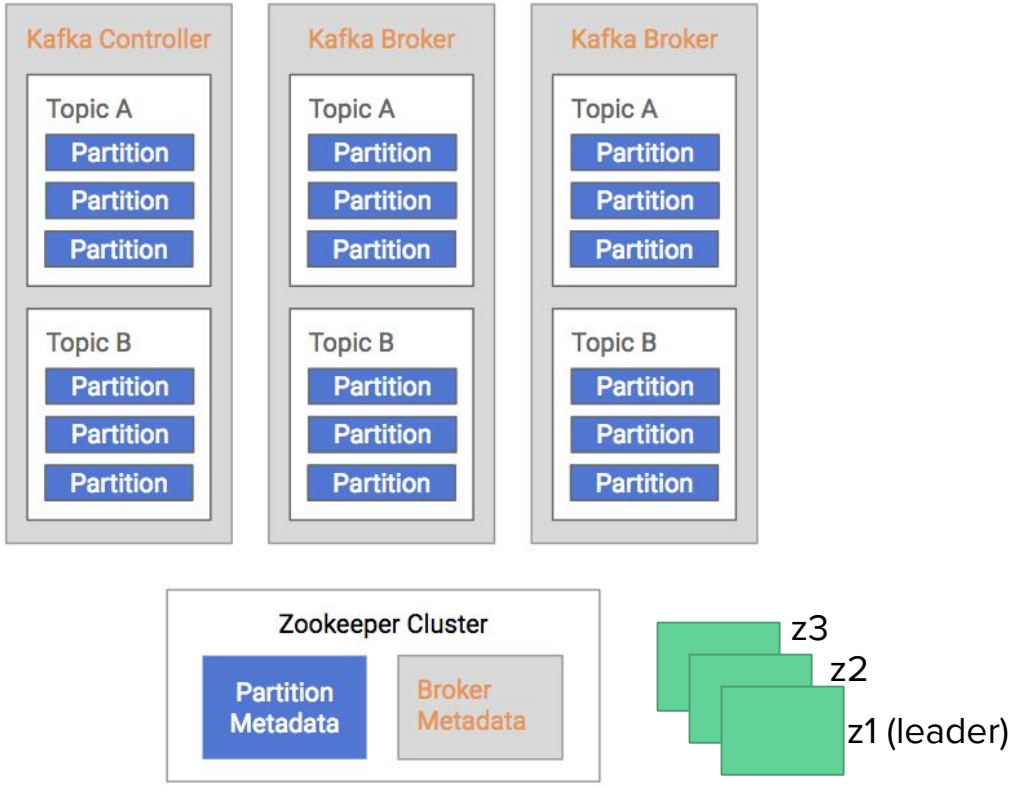
- information in ZooKeeper is organized quite similar to a file system, nodes referred to as zNodes
- zNode may act as both a file containing binary data and a directory with more zNodes as sub nodes.
- each zNode has some meta data. This metadata includes read and write permissions and version information.
- **Persistent/Sequential Znodes: nodes whose names are automatically assigned a sequence number suffix.** This suffix is strictly growing and assigned by ZooKeeper when the zNode is created. **An easy way of doing leader election with ZooKeeper is to let every server publish its information in a zNode that is both sequential and ephemeral.** Then, whichever server has the lowest sequential zNode is the leader.
- **Ephemeral Znodes**: a node that will disappear when the session of its owner ends (for discovery of hosts in your distributed system).
- possibility of **registering watchers on zNodes**: ability for clients to be notified of the next update to that zNode. With the use of watchers one can implement a message queue by letting all clients interested in a certain topic register a watcher on a zNode for that topic, and messages regarding that topic can be broadcast to all the clients by writing to that zNode.

Role of Zookeeper in Kafka Architecture

- Coordinator & manager of the brokers (keeps a list of them, manages heartbeats)
- Stores Topics, Quotas and Authorization Lists (ACL's)
- Helps in performing leader election for partitions (knowing the ISRs status)
- Sends notifications to kafka cluster in case of changes (new topic, broker dies, broker comes up, delete topics, ..)
- **Kafka cannot work without Zookeeper (momentarily)**
- By design works with an odd number of servers
- Zookeeper cluster has a leader (handles writes) and the rest of the servers are followers (handle reads)

/opt/kafka/bin/**zookeeper-shell.sh** 10.156.0.12:2181 ls /brokers/ids
/opt/kafka/bin/**zookeeper-shell.sh** 10.156.0.12:2181 ls /brokers/topics

Role of Zookeeper in Kafka Architecture



Role of Zookeeper in Kafka

In Electing a controller process. The controller (active controller) is one of the brokers and is **responsible for deciding and maintaining the leader/follower relationship for all the partitions.** When a node shuts down, it is the controller that tells which other replicas become partition leaders and tells them to replace the partition leaders on the node that is going away. Zookeeper is used to elect a controller, making sure there is only one and elect a new one it if it crashes.

Cluster membership - which brokers are alive and part of the cluster?

Topic configuration - which topics exist, how many partitions each has, where are the replicas, who is the preferred leader, what configuration overrides are set for each topic.

(0.9.0) - Quotas - how much data is each client allowed to read and write. Quota defaults and overrides are written to zookeeper and are read by all brokers immediately. Therefore, quotas are enforced on clients without the need for a restart.

(0.9.0) - ACLs - who is allowed to read and write to which topic.

Role of Zookeeper: > Kafka 0.9

ZooKeeper is no longer used by the producer or consumer in 0.9.0 (offsets commited live in a kafka topic named _consumer_offsets_) . However, Kafka stil stores much of its metadata in ZooKeeper -- **topics, partitions, replicas, the ISR, brokers, etc.** A single produce or consume does not interact with ZooKeeper. Zookeeper is involved in adding a new broker, adding a new topic, recovering from a failed broker, recovering from a failed controller, etc. Also, **brokers heartbeat with ZooKeeper, allowing ZooKeeper to know if a broker fails.**

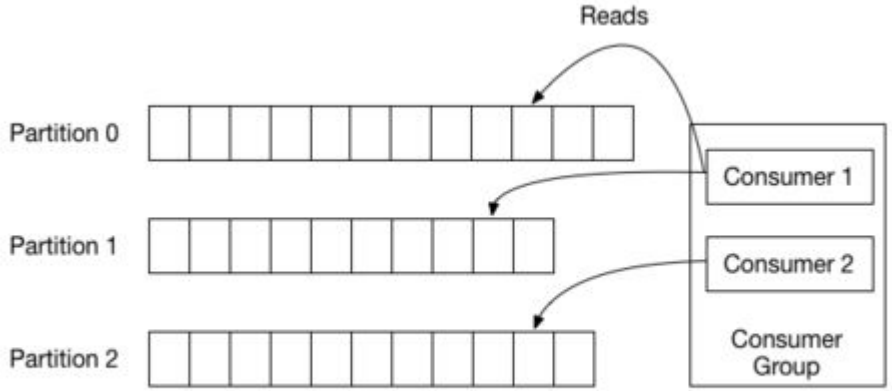


Figure 1: Consumer Group

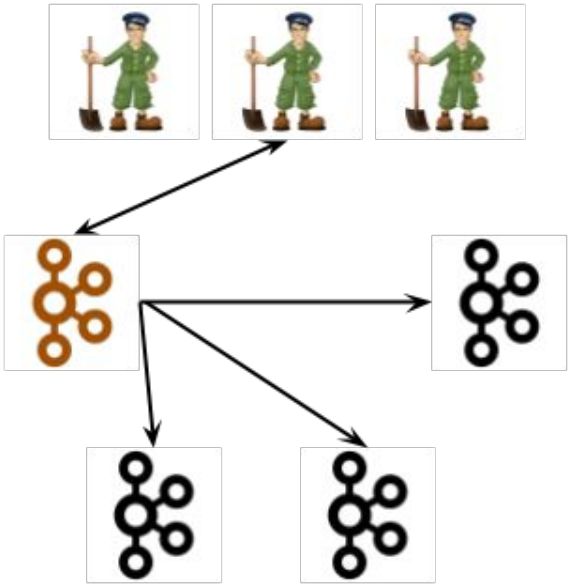
KIP 500 (no Zookeeper) - Apache Kafka 2.8 in testing

KIP 500: Replace ZooKeeper with a self-managed quorum - **Kafka 2.8. an early-access look at Kafka without ZooKeeper**

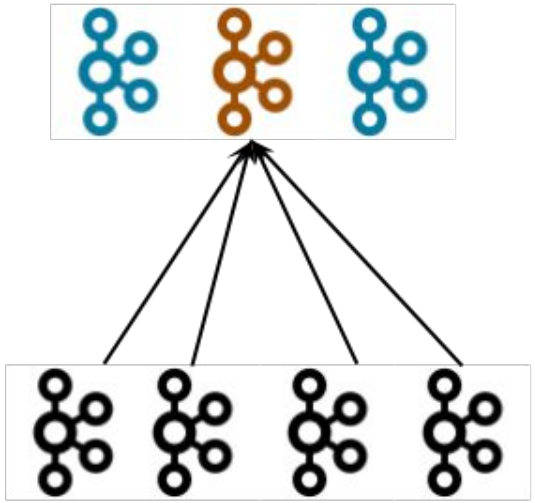
Motivation: “Currently, Kafka uses ZooKeeper to store its metadata about partitions and brokers, and to elect a broker to be the Kafka Controller. We would like to remove this dependency on ZooKeeper.”

At a high level, KIP-500 works by moving topic metadata and configurations out of ZooKeeper and into a new internal topic named **@metadata**. This topic is managed by an internal Raft quorum of “controllers” and is replicated to all brokers in the cluster. The leader of the Raft quorum serves the same role as the controller in clusters today. A node in the KIP-500 world can serve as a controller, a broker, or both, depending on the new process.roles configuration.

KIP 500 (no Zookeeper) - Apache Kafka 2.8 in testing



Current



Proposed

3 controller nodes instead of Zookeeper - RAFT quorum