# Kafka Cluster Topology

* Kafka
  + What is kafka?
    - Open-source software, developed by LinkedIn, and passed to the Apache Foundation.
    - Messaging system that collects and processes extensive amounts of data in real-time.
    - Usage: like in our case, get data from API’s and pass it to other services, control-systems that get data from sensors etc.
    - Originally designed to track the behavior of visitors to websites.
  + How kafka works?
    - “Producers” send data streams to “topics” – saved in the kafka brokers
    - Reading data from “topics” using a “Consumer”
  + Architecture
    - A kafka cluster contains 1 or more “Brokers”, for resilience, and each topic is replicated to all the brokers.
    - A “Zookeeper” is an entity that manages the kafka brokers, producers, and consumers. It is selecting the master broker and manages replication between brokers.
  + Kafka-Broker
    - A kafka cluster is made up of multiple Kafka Brokers. Each one has a unique ID. The broker contains topic log partitions.
    - Connecting to one broker bootstraps a client to the entire kafka cluster.
    - A kafka cluster can have more than 1000 brokers.
* Zookeeper
  + A software that used mainly in Distributed systems for service synchronization and naming registry
  + In kafka, zookeeper is used to track the kafka brokers status and maintain a list of topics and messages
  + Developed by Yahoo, open source under the Apache License.
  + Zookeeper is a must for kafka.
  + Zookeeper functions:
    - Controller election – choosing the leader broker in the kafka cluster.
    - Cluster membership – keeps a list of all functioning brokers in the cluster.
    - Topic configuration – maintains the configuration of all the topics, and the list of existing topics, number of partitions for each, location of replicas, preferred leader node
    - ACLs – ACLs for all the topics, includes who can read/write to each topic, list of consumer groups and their members
    - Quotas – how much data each client is allowed to read/write.
* Kafka Topic
  + Topics are the categories used to organize messages – each topic has a unique name across the Kafka-Cluster.
  + Producers write to topics and consumers read from topics.
  + Every topic is multi-subscriber – which means that multiple consumers can read at the same time.
  + topics are partitioned and replicated across brokers.
    - Each partition contains record in an unchangeable sequence.
    - each record in a partition is assigned and identified by its unique offset
    - Topic can have multiple partition logs; this allows multiple consumers to read in the same time.
    - Partitions allow topics to be parallelized by splitting the data into topic across multiple brokers.
  + Topic replication
    - Implemented at the partition level.
    - Redundant unit of a topic is called a replica, each topic usually has one or more replicas – partition contain messages that are replicated over a few kafka brokers in the cluster.
    - Every partition has one server acting as a leader and the rest as followers.
    - The leader replica handles all read/write and replicated to the followers.
    - When producer push record to a topic it is published to the leader.
  + Producer
    - Must know which partition to write to