Contents

[1 General Introduction 2](#_Toc35704534)

[1.1 2](#_Toc35704535)

[2 ZooKeeper 2](#_Toc35704536)

[3 Apache Kafka 2](#_Toc35704537)

[3.1 Source/Credits 2](#_Toc35704538)

[3.2 Important Terms 2](#_Toc35704539)

[3.3 Architecture 3](#_Toc35704540)

# General Introduction

## 

# Sources/Credits/Acknowledgements

<https://kafka.apache.org/documentation/> - Excellent introduction to Kafka

<https://jaceklaskowski.gitbooks.io/apache-kafka/kafka-brokers.html>

<https://sookocheff.com/post/kafka/kafka-in-a-nutshell/>

# ZooKeeper

# Apache Kafka

## Important Terms

|  |  |
| --- | --- |
| **Cluster** | A running Kafka Instance that clients will connect to. Consists of an ensemble of stateless Kafka brokers that host topics. The cluster state is maintained using **ZooKeeper.** |
| **Broker** | Source - <https://jaceklaskowski.gitbooks.io/apache-kafka/kafka-brokers.html>  “A **Kafka server**, a **Kafka broker** and a **Kafka node** all refer to the same concept and are synonyms  The above line is significant. Assume that a broker is a server or a node. This assumption makes it easy to understand how Kafka works.   * A Kafka broker receives messages from producers and stores them on disk keyed by unique **offset**. * A Kafka broker allows consumers to fetch messages by topic, partition and offset. * **Kafka brokers can create a Kafka cluster by sharing information between each other directly or indirectly using Zookeeper** |
| **Partitions** | * You can lay out a topic (as partitions) across a cluster of machines to allow data streams larger than the capability of a single machine. Partitions are log files on disk with sequential write only. Kafka guarantees message ordering in a partition * Partitions mirror the physical manner in which a topic gets persisted across machines. * Partitions allow you to parallelize a topic by splitting the data in a particular topic across multiple brokers — each partition can be placed on a separate machine to allow for multiple consumers to read from a topic in parallel. Note – Multiple partitions can be on the same machine. |
| **Producers** | These are third party applications that send streams of data / messages to topics in a Kafka cluster. |
| **Consumers** | These are third party applications that read messages from topics in a Kafka cluster. |
| **Offset** | Each partition is an ordered, immutable sequence of records that is continually appended to—a structured commit log. The records in the partitions are each assigned a sequential id number called the *offset* that uniquely identifies each record within the partition |
| **Streams** |  |
| **Connectors** |  |
| **Topics** | <https://jaceklaskowski.gitbooks.io/apache-kafka/kafka-brokers.html>   * **Producers will always write to a topic**      * A Kafka topic is spread across a Kafka cluster as a logical group of one or more **partitions** * **Topics store messages and are partitioned and replicated across multiple brokers** * **Topics** are virtual groups of one or many [partitions](https://jaceklaskowski.gitbooks.io/apache-kafka/kafka-topics.html#partitions) across [Kafka brokers](https://jaceklaskowski.gitbooks.io/apache-kafka/kafka-brokers.html) in a Kafka cluster. |
| **Zookeeper** | Service that provides distributed co-ordination |
| **log end offset** | The **log end offset** is the offset of the last message written to a log. |
| **high watermark offset** | The **high watermark offset** is the offset of the last message that was successfully copied to all of the log’s replicas. |
| **Replication-factor** |  |
| **Leaders** |  |
| **Followers** |  |
| **Replicas** |  |
| **Consumer groups** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Architecture

Kafka is run as a cluster on one or more servers that can span multiple datacenters

 

