Open Source Datacenter Systems

Apache KAFKA

LAMP Stack

- **L** – Linux
- **A --** Apache
- **M –** MySQL/MSQL/MariaDB
- **P** – PHP/Perl/Python

SMACK Stack

- **Spark** – A general engine for large-scale data processing, enabling analytics from SQL queries to machine learning, graph analytics, and stream processing
- **Mesos** – Distributed systems kernel that provides resourcing and isolation across all the other SMACK stack components. Mesos is the foundation on which other SMACK stack components run.
- **Akka** – A toolkit and runtime to easily create concurrent and distributed apps that are responsive to messages.
- **Cassandra** – Distributed database management system that can handle large amounts of data across servers with high availability.
- **Kafka** – A high throughput, low-latency platform for handling real-time data feeds with no data loss.

Kafka Introduction

- Kafka is an Open Source project managed by Apache Foundation (https://kafka.apache.org/)
- Originally developed at LinkedIn in 2010
- Commercial support provided by Confluent (https://www.confluent.io/)
- Competitors – RabbitMQ, ActiveMQ
- Decouples producer and consumer
- Highly Scalable – billions of messages per day
- High performance
- Non-blocking architecture
- Low latency

Kafka Features

- Apache Kafka is a distributed messaging system providing fast, highly scalable and redundant messaging through a publisher-subscriber model. Apache Kafka is highly available and resilient to node failures and supports automatic recovery.
- Apache Kafka has provision of robust queue that can handle a high volume of data and has enabler to pass on the messages from one end point to another. Kafka is suitable for both offline and online message consumption.
- Apache Kafka is built on top of the Zookeeper synchronization service. All Kafka messages are organized into topics. It integrates very well with Apache Storm and Spark for real-time streaming data analysis.

Kafka Capabilities

A streaming platform has three key capabilities:

- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
- Store streams of records in a fault-tolerant durable way.
- Process streams of records as they occur.

Kafka is generally used for two broad classes of applications:

- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of data

Kafka Design

Kafka has four core APIs:

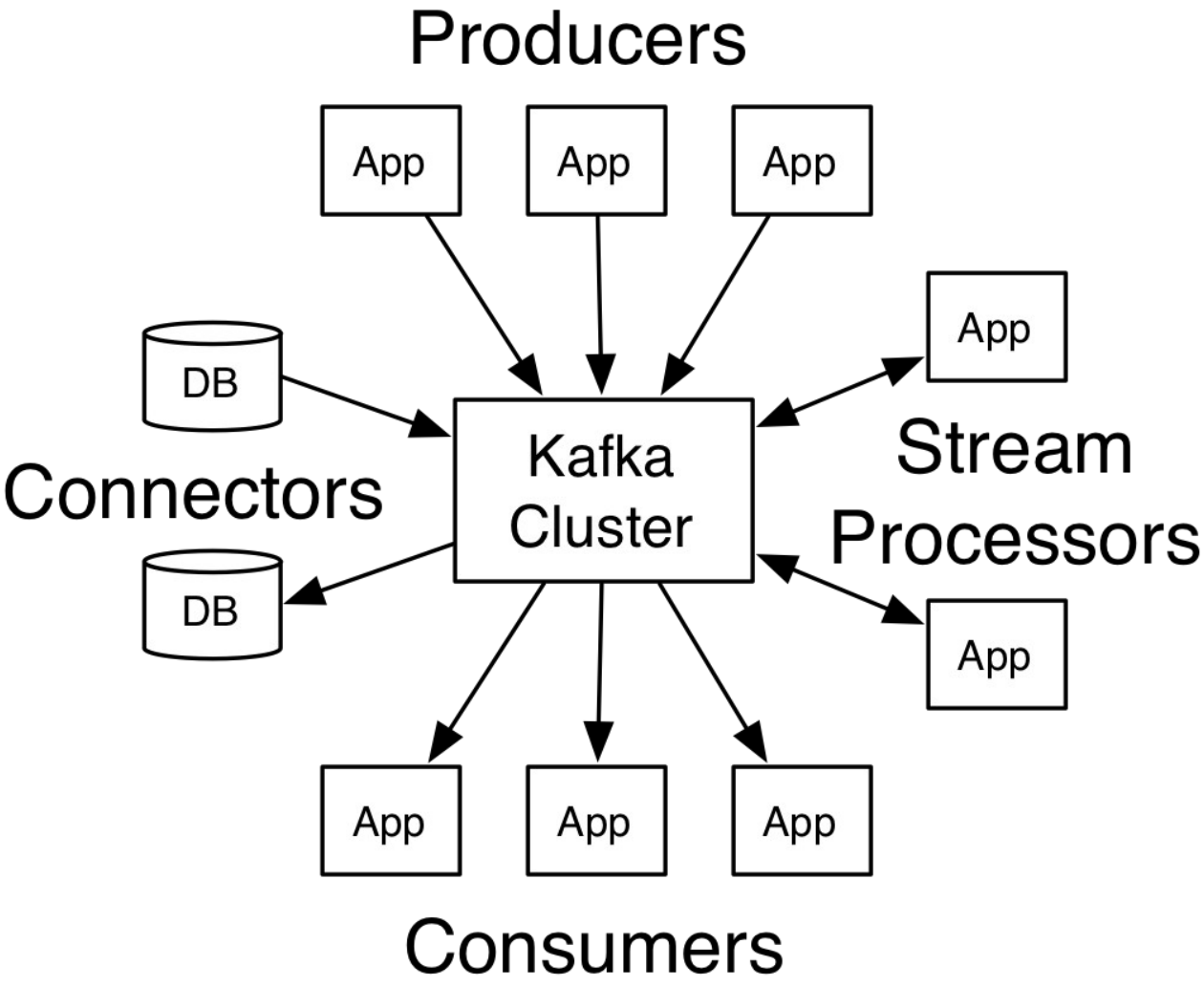
The Producer API allows an application to publish a stream of records to one or more Kafka topics.

The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them.

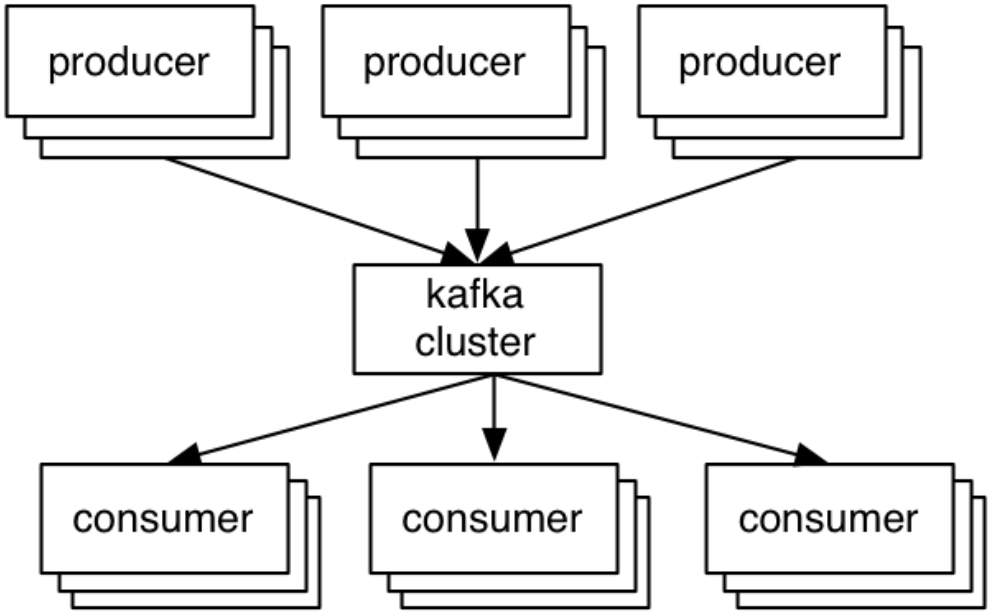
The Streams API allows an application to act as a *stream processor*, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.

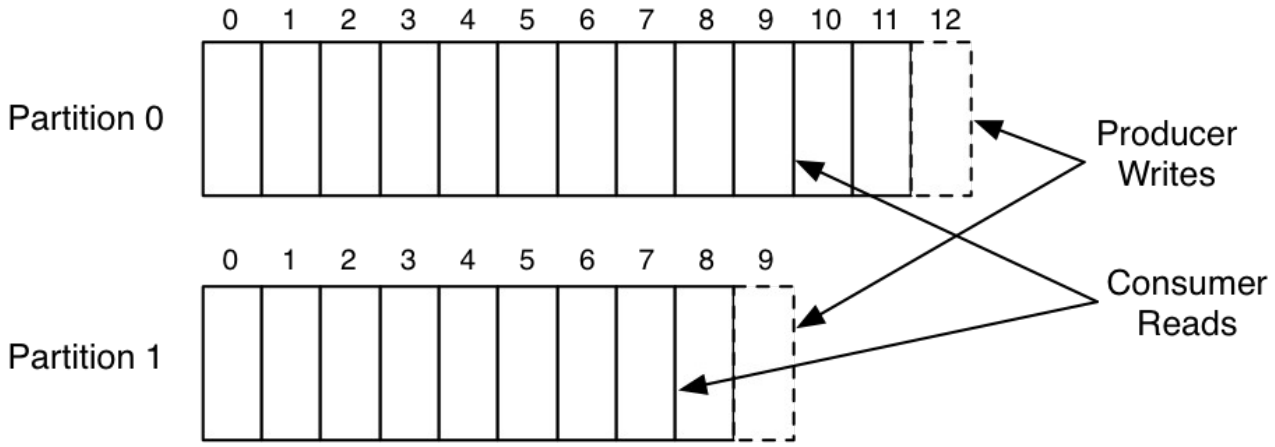
The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

Messaging Infrastructure

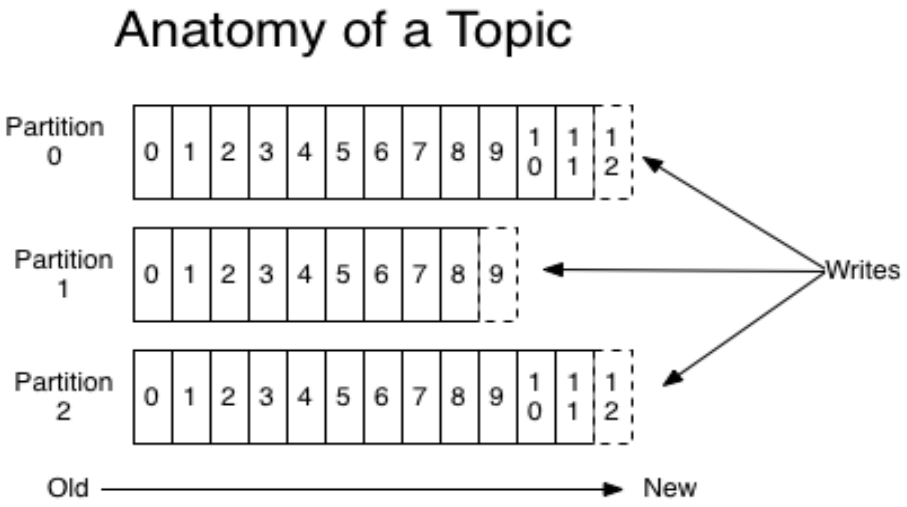


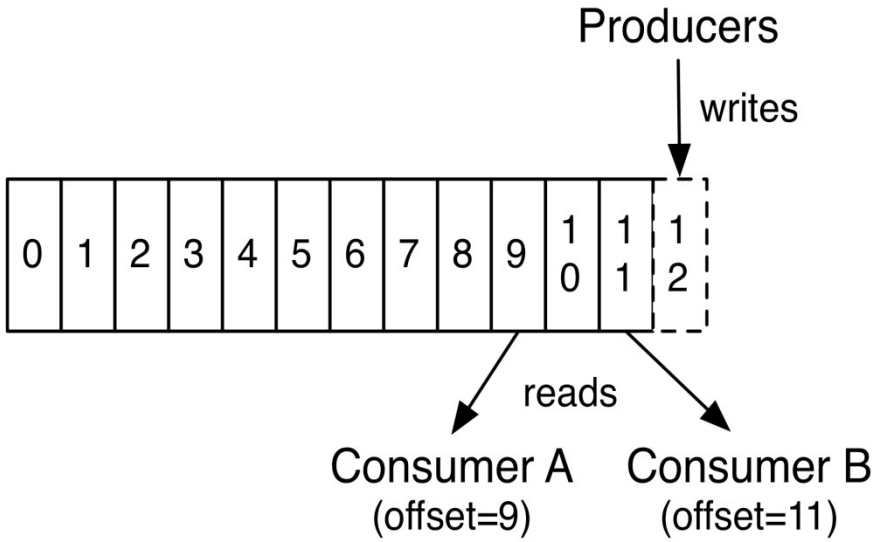
Kafka Architecture

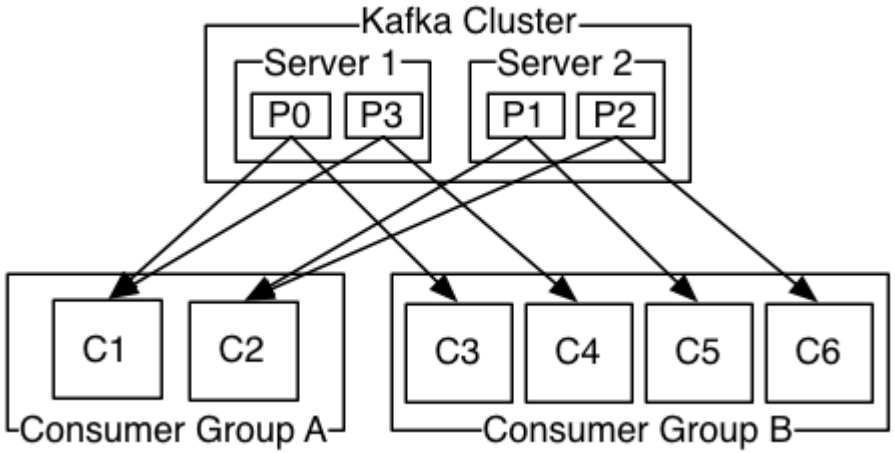




Kafka Infrastructure







Kafka Features

Multi-tenancy

You can deploy Kafka as a multi-tenant solution. Multi-tenancy is enabled by configuring which topics can produce or consume data. There is also operations support for quotas. Administrators can define and enforce quotas on requests to control the broker resources that are used by clients.

Guarantees

At a high-level Kafka gives the following guarantees:

Messages sent by a producer to a particular topic partition will be appended in the order they are sent. That is, if a record M1 is sent by the same producer as a record M2, and M1 is sent first, then M1 will have a lower offset than M2 and appear earlier in the log.

A consumer instance sees records in the order they are stored in the log.

For a topic with replication factor N, we will tolerate up to N-1 server failures without losing any records committed to the log.

Kafka Features

Kafka for Stream Processing

It isn't enough to just read, write, and store streams of data, the purpose is to enable real-time processing of streams.

In Kafka a stream processor is anything that takes continual streams of data from input topics, performs some processing on this input, and produces continual streams of data to output topics

Kafka as a Storage System

Any message queue that allows publishing messages decoupled from consuming them is effectively acting as a storage system for the in-flight messages. What is different about Kafka is that it is a very good storage system

Tibco Kafka

- TIBCO ActiveMatrix BusinessWorks Plug-in for Apache Kafka – Community Edition plugs into TIBCO ActiveMatrix BusinessWorks.
- Apache Kafka palette can be used to create producers, consumers and perform send message and receive message operations.
- The plug-in provides the main features given below:
- **Kafka Connection Shared Resource**
  Kafka connection shared resource is used to connect and fetch the list of topics from Kafka server. The shared resource is used by the other activities for configuring the connection.
  Kafka SendMessage Activity
- This activity sends message to the Kafka consumer. This activity performs as Kafka producers which sends the message to a specified topic and consumer can fetch the message from the specified topics.
  Kafka ReceiveMessage Activity
- This activity is a process starter activity that starts the process based on the receipt of Kafka messages.

Key Takeaways

- Apache Kafka is a very powerful messaging application
- Allows connectivity between producer and consumer
- Multiple consumers can access the same data without any performance penalty
- Kafka supports both Queueing and Publish-Subscribe concepts
- Provides replication thereby increasing resilience to failures
- Kafka can provide storage of data from bytes to Tbytes
- Kafka is able to provide real-time processing of events
- Use cases include messaging, log aggregation, monitoring and stream processing