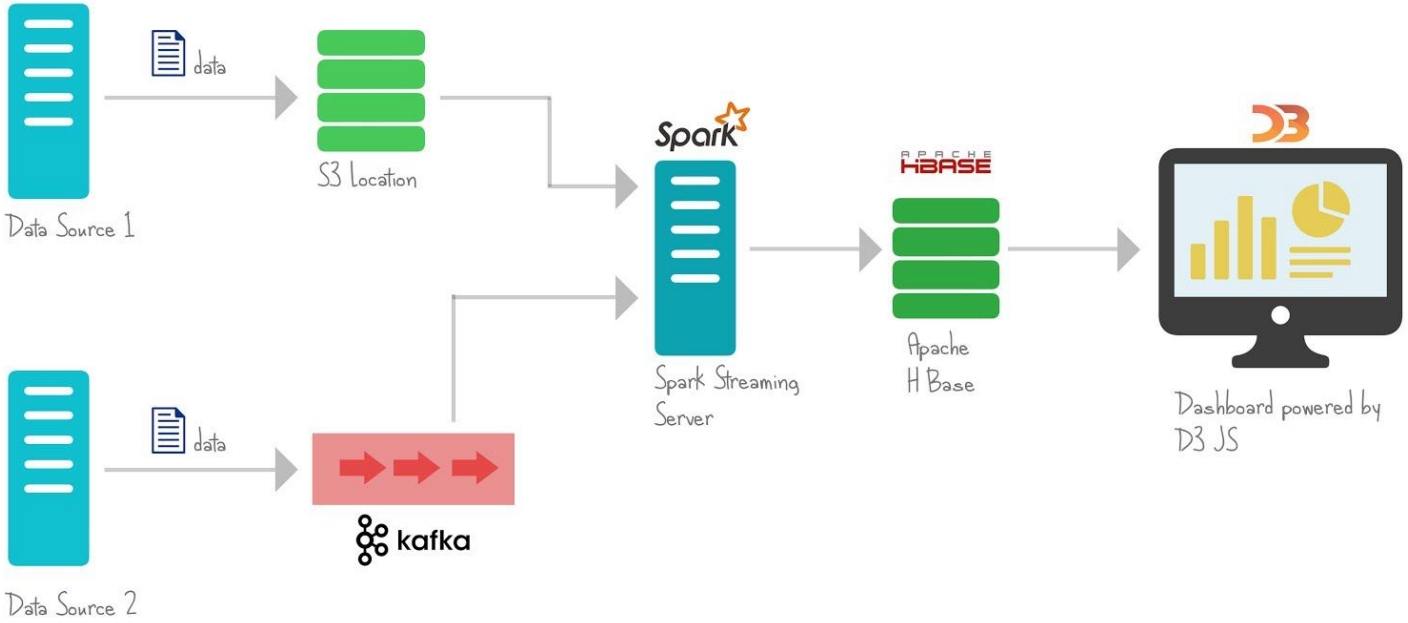
Machine Learning & Deep Learning Kafka, Spark, HBase

...

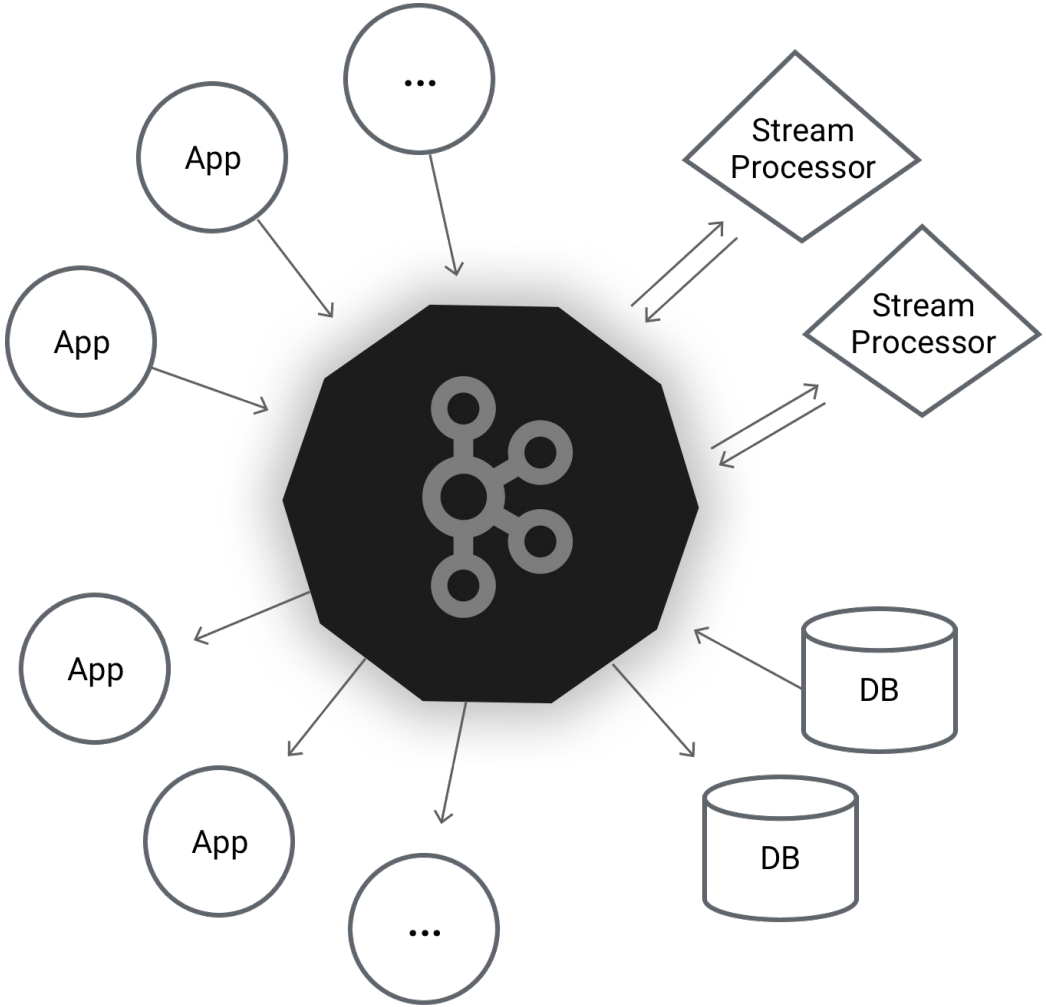
Dinesh Kumar Subramanian
Founder, DTeam


data
S3 location
Data Source 1
Spark
Spark Streaming
Server
APACHE
HBASE
Apache
H Base
D3
Dashboard powered by
D3 JS
data
kafka
Data Source 2

Today's Agenda

- Apache Kafka - 80%
  - What it is?
  - Architecture
  - Role in Pipeline
- Apache Spark - 10%
  - What it is?
  - Architecture
  - Role in Pipeline
- HBase - 10%
  - What it is?
  - Architecture
  - Role in Pipeline

What is Kafka


...
App
Stream Processor
App
Stream Processor
App
DB
App
DB
...

What is Kafka

Distributed Streaming Platform

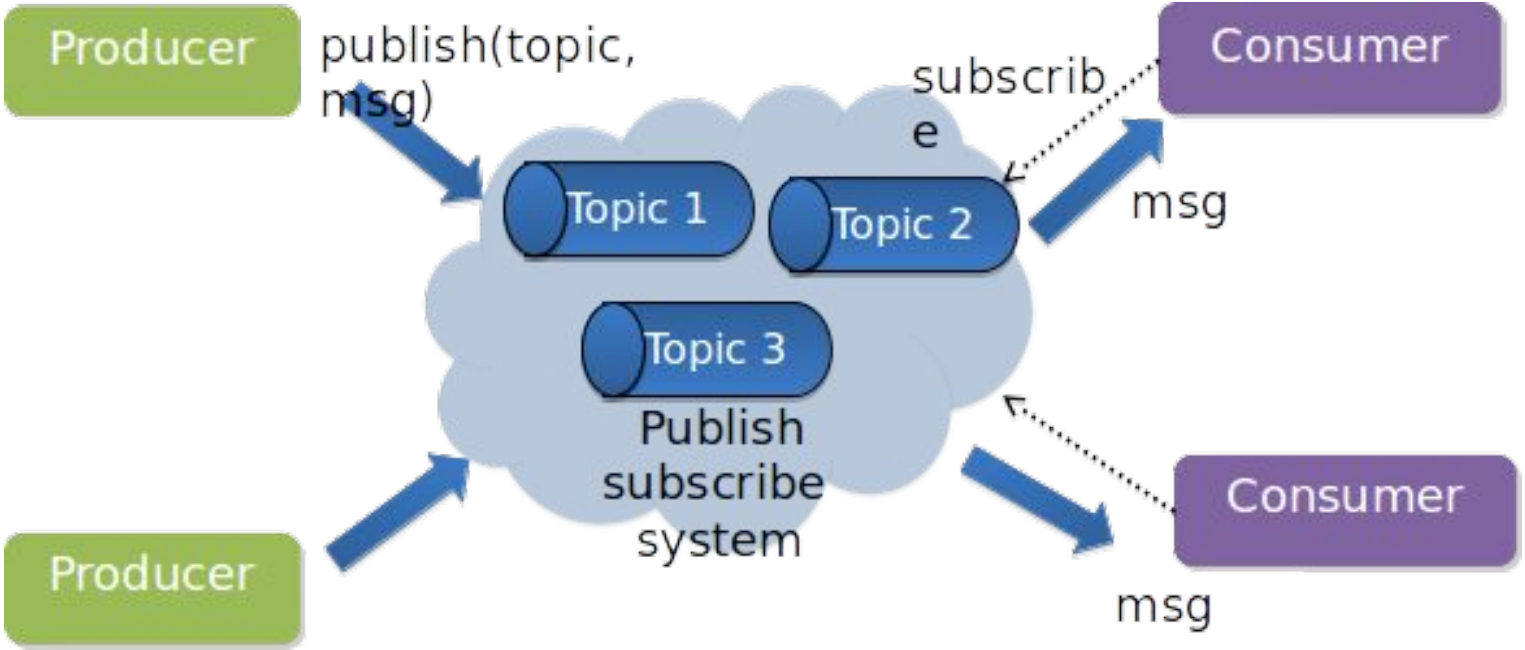
Streaming Platform ?

- 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.

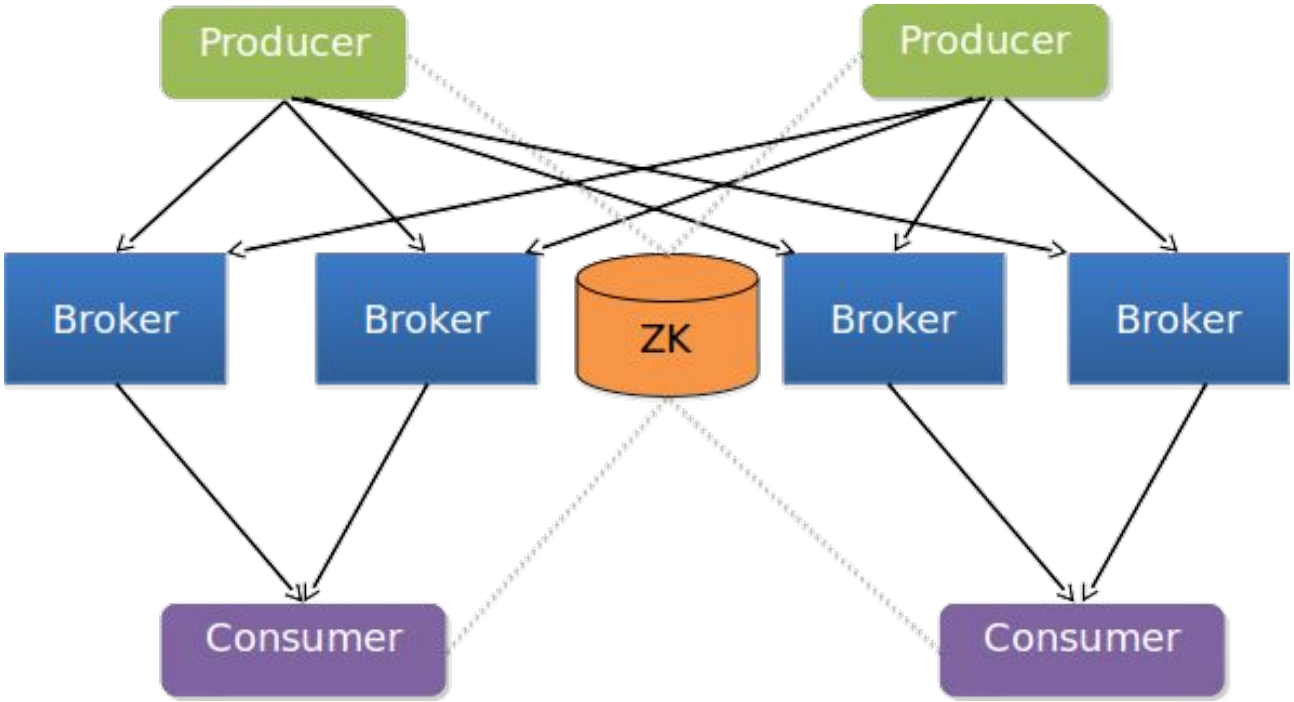
Kafka Use Cases

- 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

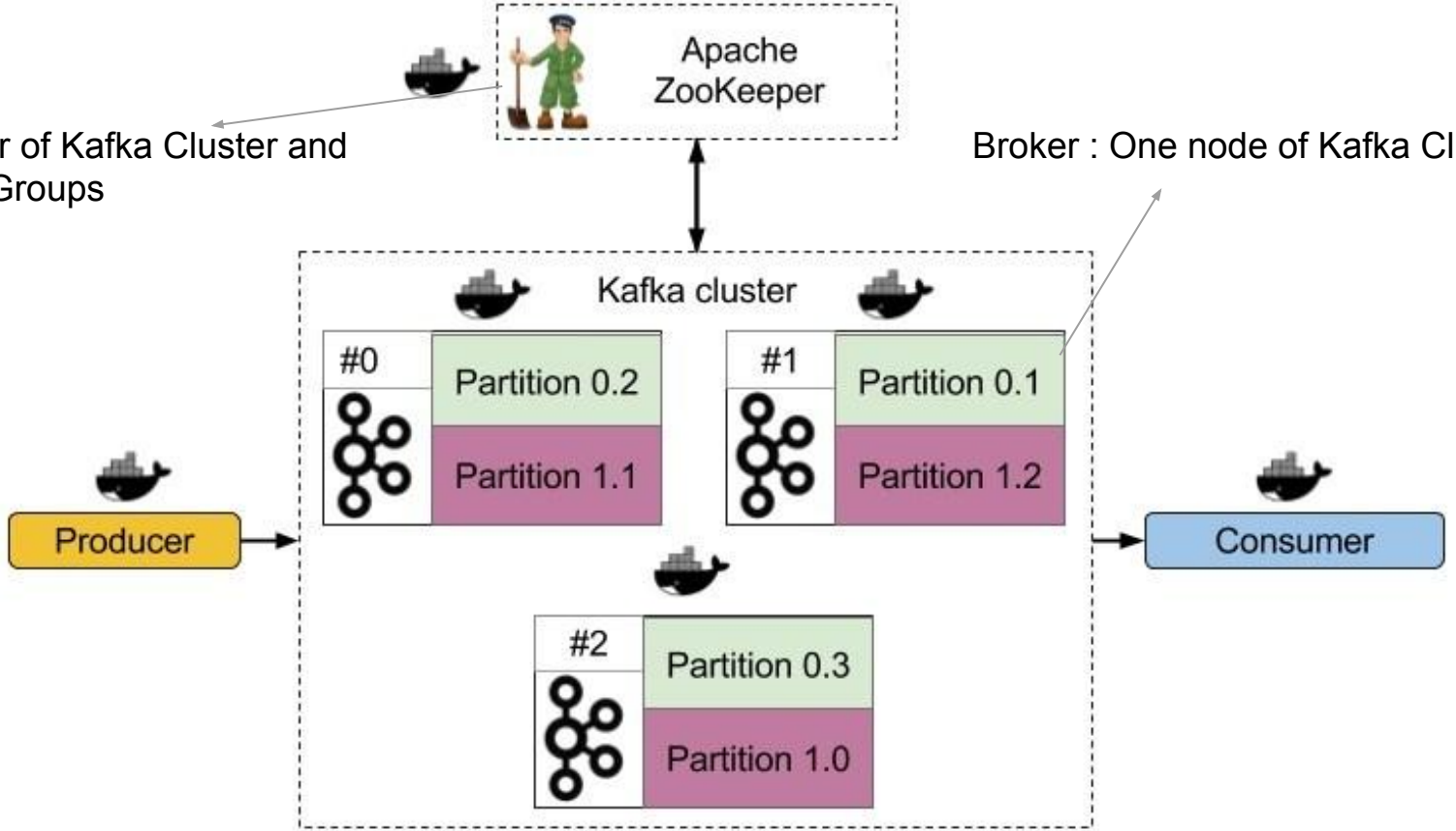
Publish-Subscribe



Architecture of Kafka



Kafka Architecture

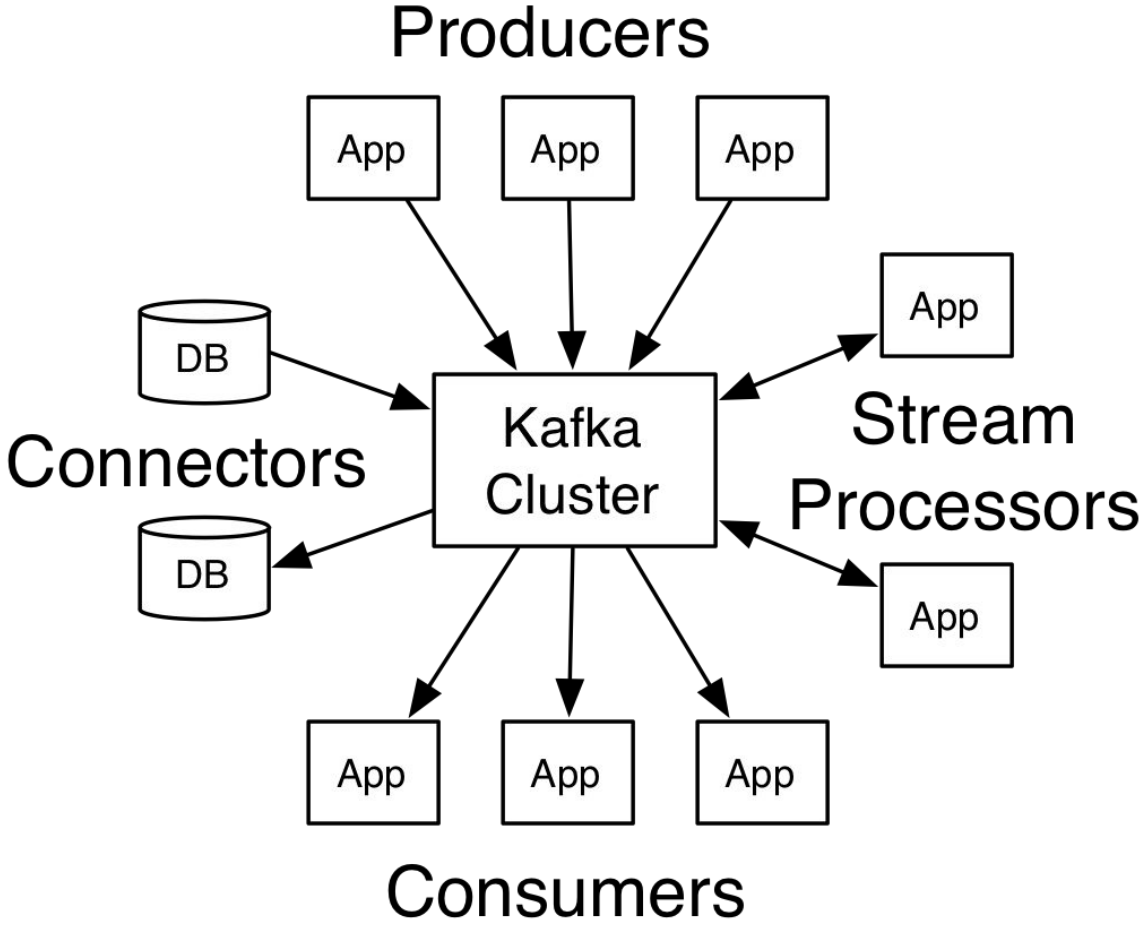


Why Apache kafka?

- Kafka is a unified platform for handling all the real-time data feeds
- Supports low latency message delivery and gives guarantee for fault tolerance in the presence of machine failures
- Has the ability to handle a large number of diverse consumers
- Is very fast, performs 2 million writes/sec
- Kafka persists all data to the disk, which essentially means that all the writes go to the page cache of the OS (RAM)
- This makes it very efficient to transfer data from page cache to a network socket

Kafka Four Core APIs

- Producer API
- Consumer API
- Streams API
- Connector API

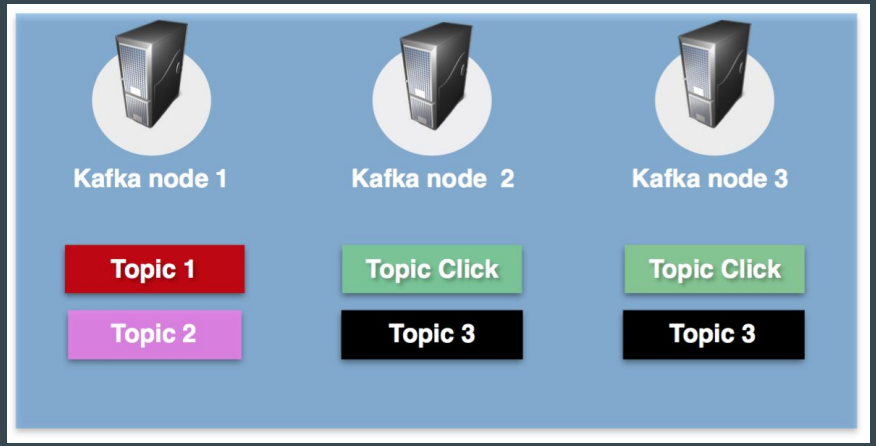


Overview

- **Topic:** A Topic is a category/feed name to which messages are stored and published.
- Topic partition: Kafka topics are divided into a number of partitions, which allows you to split data across multiple brokers.
- **Replicas:** A replica of a partition is a "backup" of a partition. Replicas never read or write data. They are used to prevent data loss.
- **Offset:** The offset is a unique identifier of a record within a partition. It denotes the position of the consumer in the partition.
- **Node:** A node is a single computer in the Apache Kafka cluster.
- **Cluster:** A cluster is a group of nodes i.e., a group of computers.

Kafka Topic?

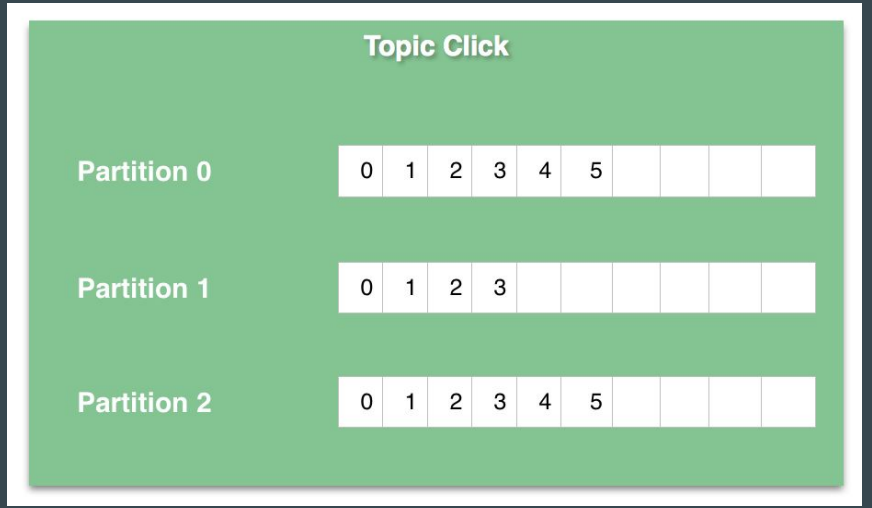
- A **Topic** is a category/feed name to which messages are stored and published.
- **Messages** are byte arrays that can store any object in any format.
- **Producer applications** write data to topics and **consumer applications** read from topics.



- Messages published to the cluster will stay in the cluster until a configurable **retention period** has passed by.
- Kafka retains all messages for a set amount of time, and therefore, **consumers are responsible to track their location.**

Topic Partition?

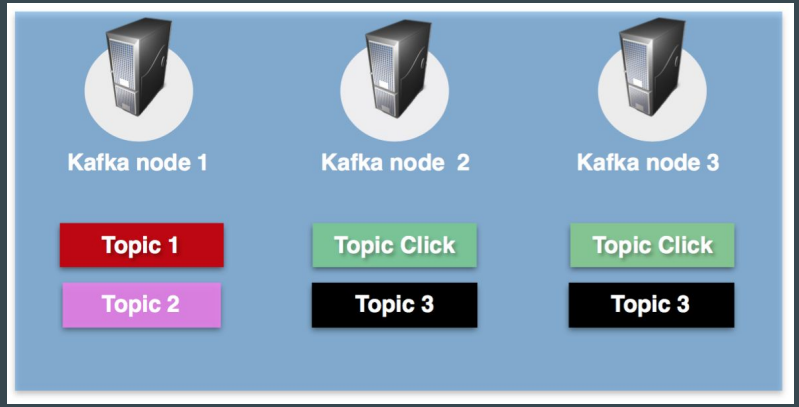
- Kafka topics are divided into a number of partitions, which contains messages in an unchangeable sequence.
- Each message in a partition is assigned and identified by its unique offset.
- A topic can also have multiple partition logs like the click-topic has in the image to the right.

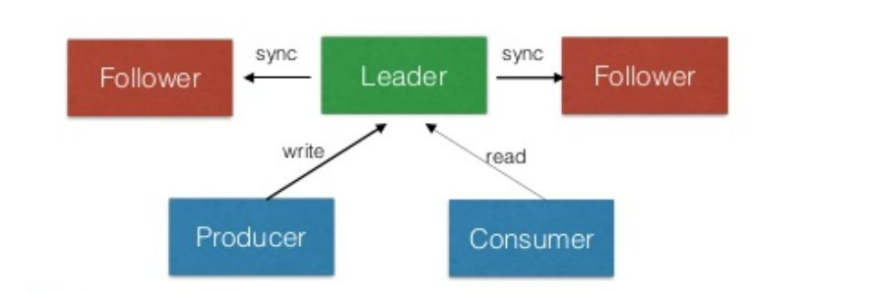


- Partitions allow you to parallelize a topic by splitting the data in a particular topic across multiple brokers.

Kafka Replicas

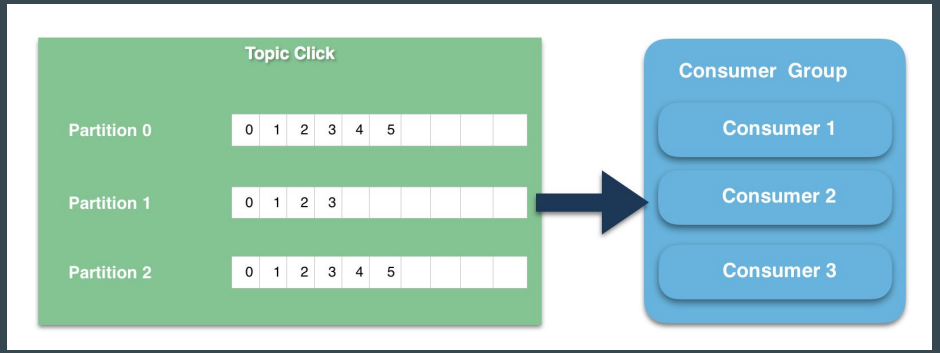
- The redundant unit of a topic partition is called a replica.
- Every partition (replica) has one server acting as a leader and the rest of them as followers.
- All Reads and Writes go through the leader of the partition





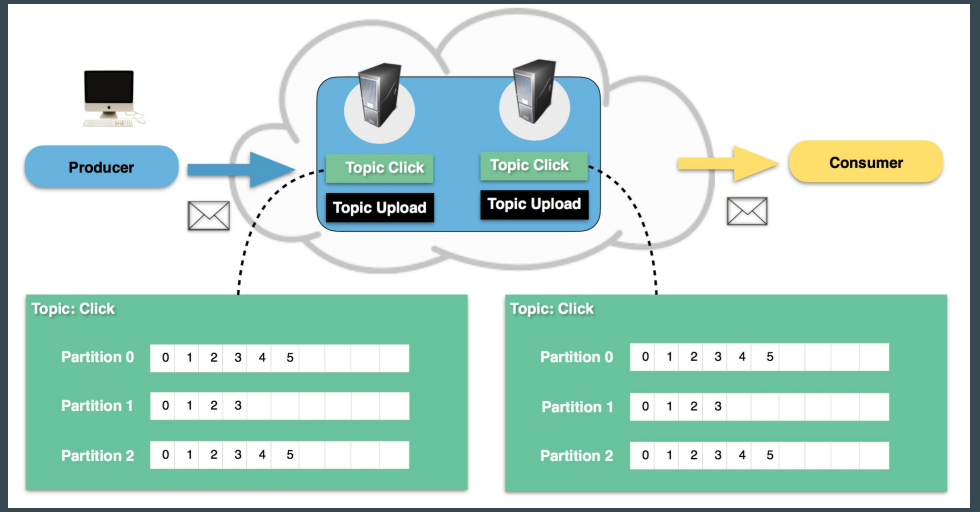
Consumer & Consumer Groups

- Consumers can read messages starting from a specific offset and are allowed to read from any offset point they choose.
- Consumers can join a group called a consumer group. A consumer group includes the set of consumer processes that are subscribing to a specific topic.
- By using consumer groups, consumers can be parallelised so that multiple consumers can read from multiple partitions on a topic



Example - Website Activity Tracking

- A user with user-id 0 clicks on a button on the website.
- The web application publishes a message to partition 0 in topic "click".
- The message is appended to its commit log and the message offset is incremented.
- The consumer can pull messages from the click-topic and show monitoring usage in real-time, or it can replay previously consumed messages by setting the offset to an earlier one.



What is Zookeeper?

- An open source, High performance distributed coordination service.
- Service used by a cluster (group of nodes) to coordinate between themselves and maintain shared data with robust synchronization techniques
- Kafka uses Zookeeper to store meta information about Kafka Cluster and Consumer client
- ZooKeeper service is mainly used to notify producer and consumer about the presence of any new broker in the Kafka system or failure of the broker in the Kafka system.
- For zookeeper to run, Java is essential.

Set up an Apache Kafka instance

- download and install Apache Kafka and Zookeeper

Installing and running kafka

```
wget http://www-us.apache.org/dist/kafka/0.10.1.0/kafka_2.10-0.10.1.0.tgz
tar -xf kafka_2.10-0.10.1.0.tgz
rm kafka_2.10-0.10.1.0.tgz
mv kafka_2.10-0.10.1.0/ kafka
cd kafka
```

- To start Zookeeper:
  - bin/zookeeper-server-start.sh config/zookeeper.properties

- To start Kafka:
  - bin/kafka-server-start.sh config/server.properties

Installing and running kafka

- Creating a Kafka Topic:

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1
--partitions 1 --topic topic-name
```

- List of Topics:

```
bin/kafka-topics.sh --list --zookeeper localhost:2181
```

- Start Producer to Send Messages

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic topic-name
```

- The producer will wait on input from stdin and publishes to the Kafka cluster. By default, every new line is published as a new message

Installing and running kafka

- Start Consumer to Receive Messages

```
bin/kafka-console-consumer.sh --zookeeper localhost:2181 –topic topic-name
--from-beginning
```

- Python Integration for kafka:

```
>>> pip install kafka-python
```

Apache Kafka Demo

What is Apache Spark?

- A general-purpose computing engine, in memory framework
- It lets you execute real-time and batch work in a scripting manner in a variety of languages with powerful fault tolerance
- Spark is designed to work with an external cluster manager or its own standalone manager
- Spark also relies on a distributed storage system to function from which it calls the data it is meant to use

Spark Context

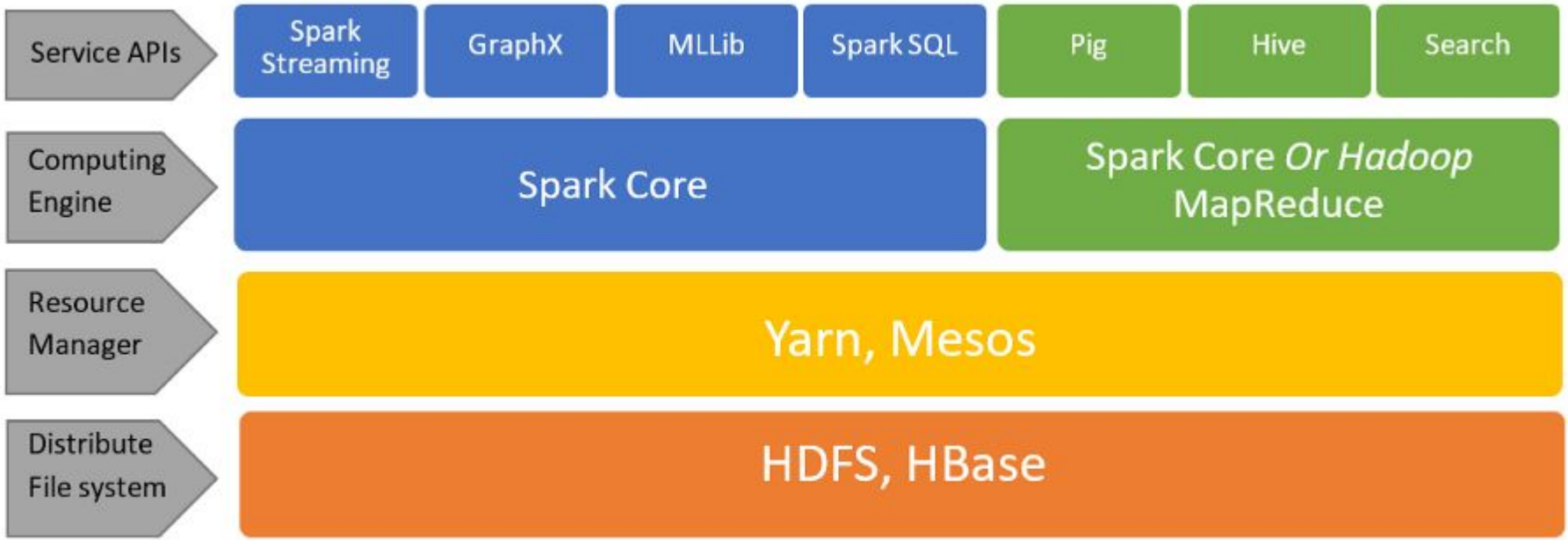


Figure 1 - Spark Context

Basic Architecture

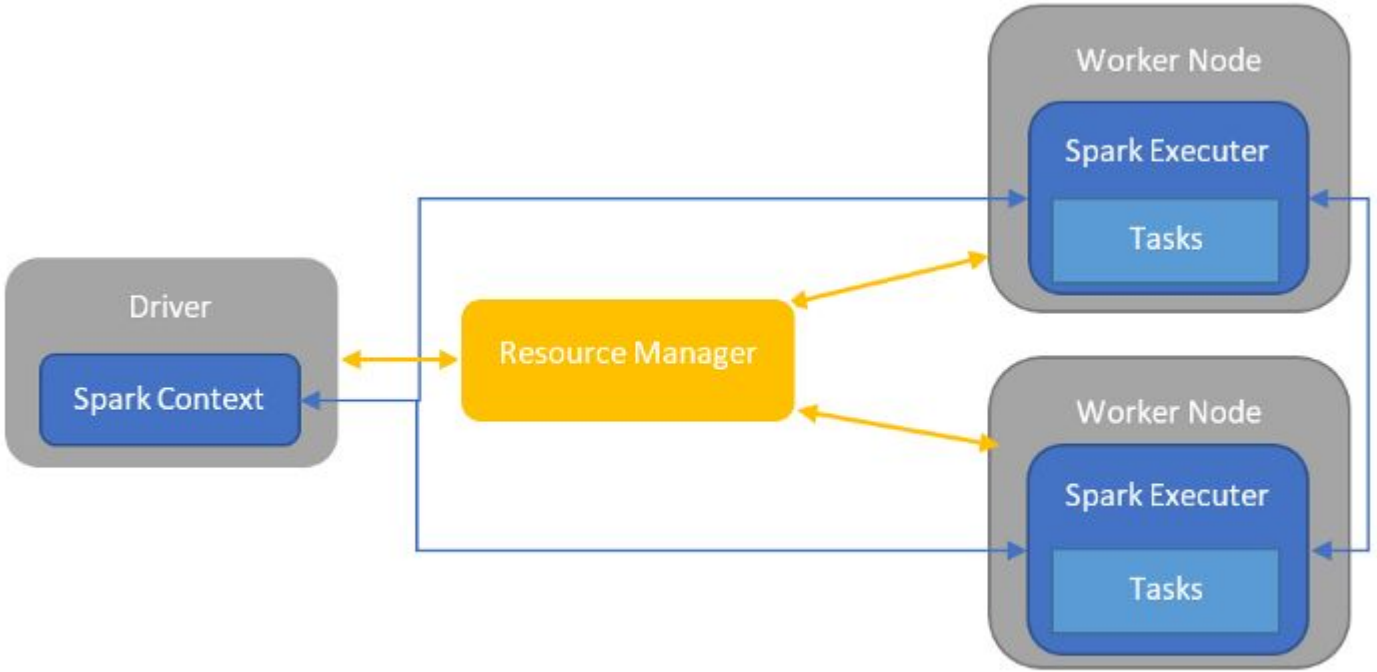


Figure 2 - Basic Architecture

Why Apache Spark?

- To put it bluntly, it has addressed many of the shortcomings Hadoop MapReduce has.
- Is roughly 10 to 100-fold faster than Hadoop MapReduce.
- Spark is a big deal in Data Science
- Some notable organisations that use Spark are:
  - Amazon,
  - NASA Jet Propulsion Labs,
  - IBM
  - Hitachi

Supported Systems of Apache Spark

- Cluster Managers:
  - Spark Standalone Manager
  - Hadoop YARN
  - Apache Mesos
- Distributed Storage Systems:
  - Hadoop Distributed File System (HDFS)
  - MapR File System (MapR-FS)
  - Cassandra
  - OpenStack Swift
  - Amazon S3
  - Kudu

Installing and Running Spark

- Check to see if Java is already installed by typing:

```
java -version
```

- Go to https://spark.apache.org/downloads.html and download a pre-built for Hadoop 2.7 version of Spark
- Then Extract it and enter the spark directory:

```
tar xvf spark-2.0.2-bin-hadoop2.7.tgz

cd spark-2.0.2-bin-hadoop2.7.tgz
```

Installing and Running Spark

- For kafka streaming integration, one need to use jar file of spark-kafka-streaming.
- Download the jar file from:
  - http://search.maven.org/remotecontent?filepath=org/apache/spark/spark-streaming-kafka-0-8-assembly_2.11/2.2.0/spark-streaming-kafka-0-8-assembly_2.11-2.2.0.jar
- And move the file to jars directory inside spark folder.
- There are two approaches for integrating Spark with Kafka:
  - Receiver-based and
  - Direct (No Receivers)
- Here we will try Direct method.
- Following code is the direct method to receive the kafka stream.

Installing and Running Spark

```
import sys

from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

if __name__ == “__main__”:
    sc = SparkContext(appName=”PythonStreamingDirectKafkaWordCount”)
    ssc = StreamingContext(sc, 2)

    brokers, topic = sys.argv[1:]
    kvs = KafkaUtils.createDirectStream(ssc, [topic],
{“metadata.broker.list”: brokers})
    lines = kvs.map(lambda x: x[1])
    counts = lines.flatMap(lambda line: line.split(“ “)) \
                  .map(lambda word: (word, 1)) \
                  .reduceByKey(lambda a, b: a+b)

    counts.pprint()
    ssc.start()
    ssc.awaitTermination()
```

Installing and Running Spark

- To start the streaming, execute the Following command:
  - bin/spark-submit --jars jars/spark-streaming-kafka-0-8-assembly_2.11-2.2.0.jar spark_test.py localhost:9092 new_topic
- If you produce message from kafka, you could see it in the Spark output:

```
-------------------------------------------
Time: 2017-01-19 13:04:10
-------------------------------------------
(u'this', 1)
(u'a', 1)
(u'test', 1)
(u'is', 1)
(u'just', 1)
```

Demo

Twitter (#twitter) → Kafka → Hbase → Spark → Hive/HBase