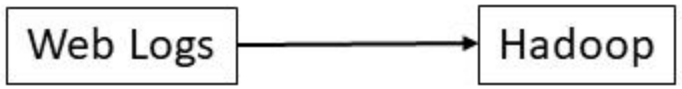
Integration with Kafka

What is Kafka?

- Data Streaming tool originally created by LinkedIn
- Open-sourced, and now used by numerous companies, including Netflix, Goldman Sachs, Airbnb, and many others.
- Can use receiver or receiver-less methods for connecting Kafka and Spark

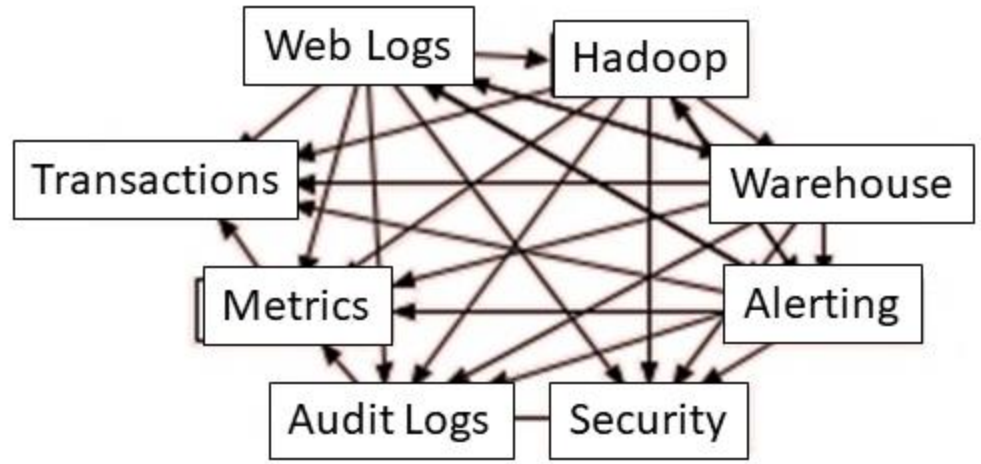
APACHE kafka®
A distributed streaming platform

Why Kafka? Increasing Complexity of distributed systems



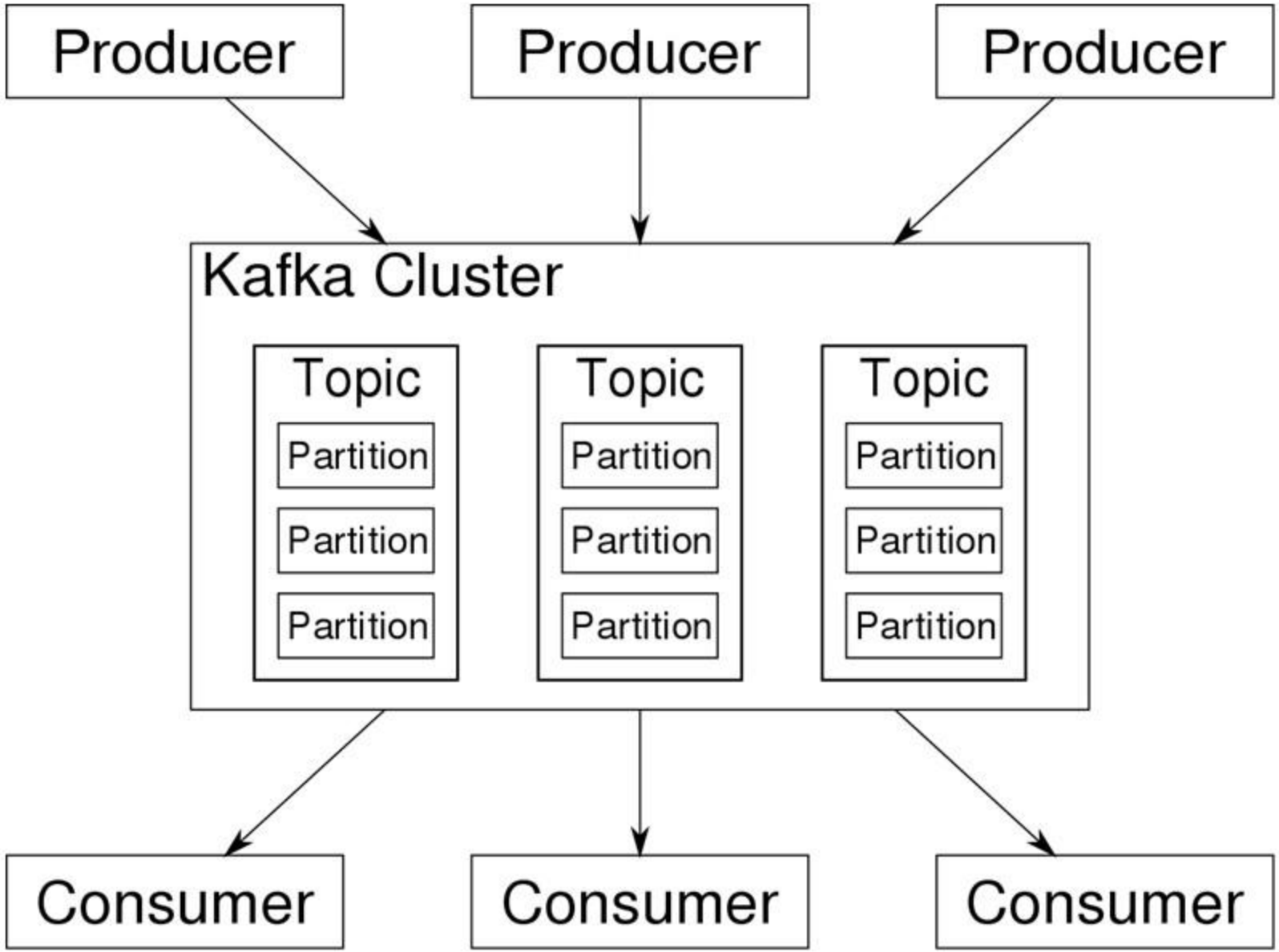
Connections = O(1)

2008



Connections = O(Systems^2)

2018


Producer
Producer
Producer
Kafka Cluster
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Consumer
Consumer
Consumer

Steps to Using Kafka In Applications

Linking

- For Python applications, you will have to add the below library and its dependencies when deploying an application

```
groupId = org.apache.spark
artifactId = spark-streaming-kafka-0-8-2.11
version = 2.2.0
```

- Full details are given in the Spark Programming guide at https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html#linking

Steps to Using Kafka In Applications

Programming

- In the streaming application code, import kafkaUtils and create an output Dstream.
- You can also specify the key and value classes and their corresponding decoder classes using variations of createStream

```
from pyspark.streaming.kafka import KafkaUtils


kafkaStream = KafkaUtils.createStream(streamingContext,[ZK quorum],
[consumer group id], [per-topic number of Kafka partitions to consume])
```

Steps to Using Kafka In Applications

Deployment

- `spark-submit` is used to launch application
- For Python applications which lack SBT/Maven project management, `spark-streaming-kafka-0-8_2.11` and its dependencies can be directly added to `spark-submit` using `--packages`

Things to Remember

- Multiple Kafka input DStreams can be created with different groups and topics for parallel receiving of data using multiple receivers.
- Kafka partitions and RDD partitions in Spark don't always correlate. Increasing # of topic-specific partitions in `KafkaUtils.createStream()` only increases the number of threads using which topics that are consumed within a single receiver.
  - Doesn't actually increase parallelism
- If you have enabled Write Ahead Logs with a replicated file system like HDFS, the received data is already being replicated in the log.