Kafka

Definition

- Kafka is a distributed publish-subscribe messaging system that is designed to be fast, scalable, and durable.
- Kafka maintains feeds of messages in topics.
- Producers write data to topics and consumers read from topics.
- Since Kafka is a distributed system, topics are partitioned and replicated across multiple nodes.

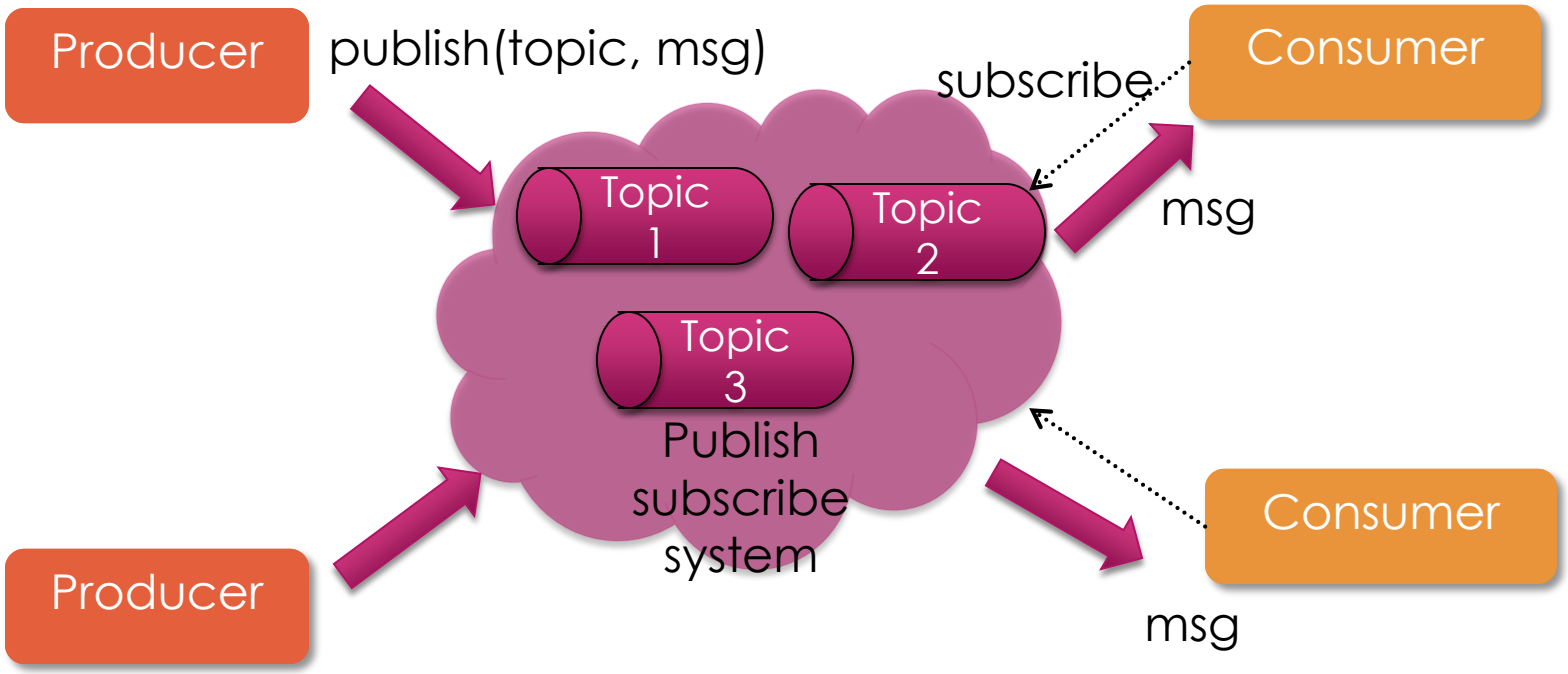
Definition

- Messages are simply byte arrays and one can use them to store any object in any format.
- It is possible to attach a key to each message, in which case the producer guarantees that all messages with the same key will arrive to the same partition.
- When consuming from a topic, it is possible to configure a consumer group with multiple consumers.
- Each consumer in a consumer group will read messages from a unique subset of partitions in each topic they subscribe to, so each message is delivered to one consumer in the group, and all messages with the same key arrive at the same consumer.

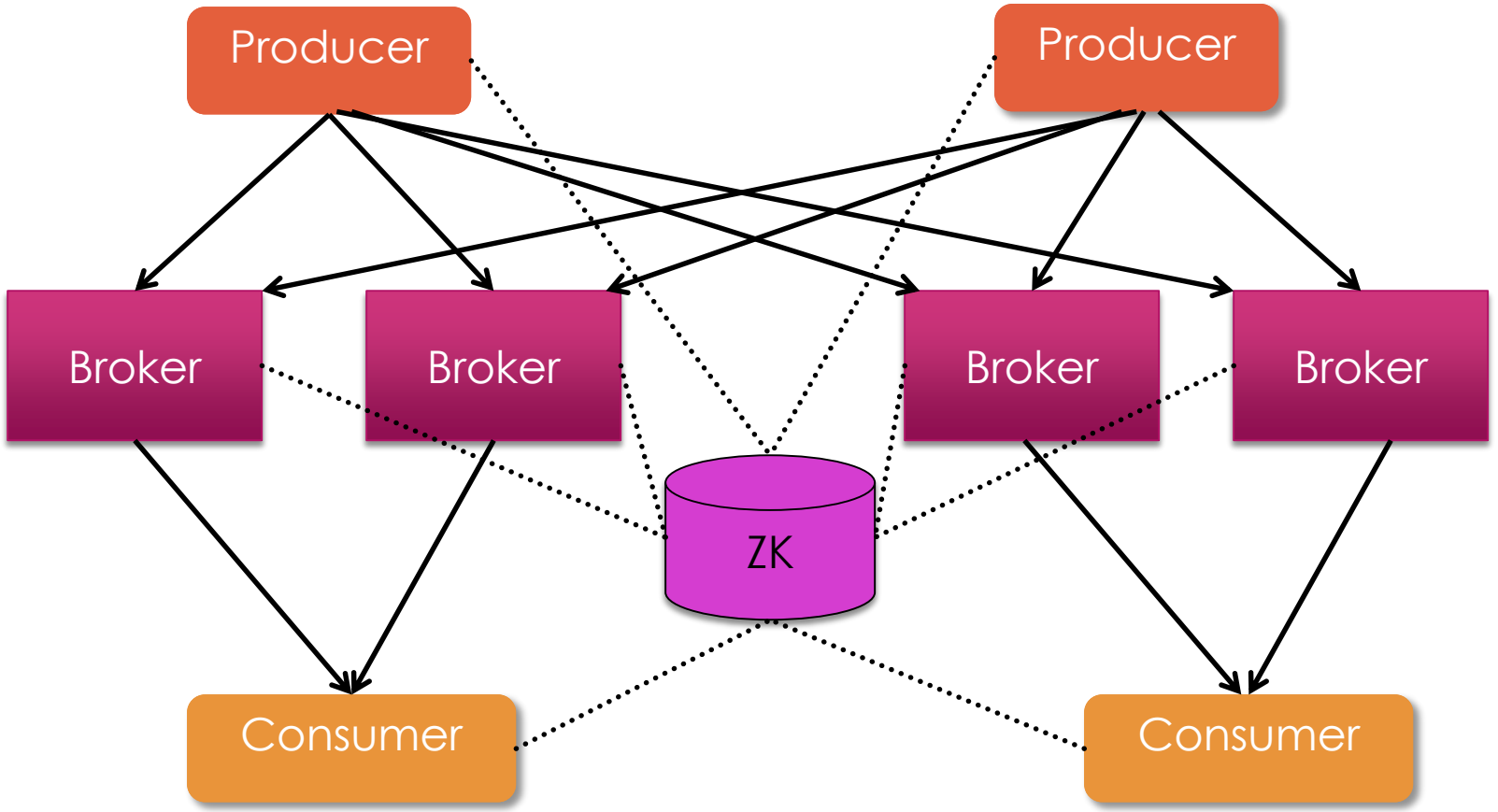
Some Use-Cases

- **Website activity tracking**: The web application sends events such as page views and searches Kafka, where they become available for real-time processing, dashboards and offline analytics in Hadoop
- **Operational metrics:** Alerting and reporting on operational metrics. One particularly fun example is having Kafka producers and consumers occasionally publish their message counts to a special Kafka topic; a service can be used to compare counts and alert if data loss occurs.
- **Log aggregation**: Kafka can be used across an organization to collect logs from multiple services and make them available in standard format to multiple consumers, including Hadoop and Apache Solr.
- **Stream processing**: A framework such as Storm reads data from a topic, processes it and writes processed data to a new topic where it becomes available for users and applications. Kafka's strong durability is also very useful in the context of stream processing.

What is pub sub ?



Architecture



Installation

- Download Binary from http://kafka.apache.org/downloads.html
- Unzip and set path to bin (if on Linux), set path to bin/windows (if on windows)
- Modify configuration (config/server.properties), to start off with the keys that require modification are.

```
log.dirs. (point to a directory which would be used to persist messages)

zookeeper.connect (point it to a ZK end point)
```

- **Start zookeeper**
- Start Kafka

```
kafka-server-start.<sh|bat> <path to configDir>/server.properties
```

Create topics and consume messages

- Create a topic

```
kafka-topics.<sh|bat>  --create --topic NEWS --zookeeper localhost:2181 --partitions 2 --
replication-factor 1
```

- Describe the newly created topic

```
kafka-topics.<sh|bat>  --describe --topic NEWS --zookeeper localhost:2181
```

- Use the shipped producer to send some test messages

```
kafka-console-producer.<bat|sh> --broker-list "localhost:9092" --topic NEWS
```

- Use the shipped consumer to read the messages

```
kafka-console-consumer.<bat|sh> --topic NEWS --zookeeper localhost:2181
```

Multiple brokers

- You can choose to run multiple brokers, each on separate node.
- If you want to simulate multiple brokers running on the same host, do the below

  Create separate data directories (say E:/tmp/brokerdata/0, E:/tmp/brokerdata/1, E:/tmp/brokerdata/2)

  Create 3 server.properties, say server0.properties, server1.properties and server2.properties with the following content (replace X in the below with 0/1/2).

```
broker.id=X
listeners=PLAINTEXT://:909X
log.dirs=E:/tmp/brokerdata/X
zookeeper.connect=localhost:2181
```

- Start 3 bokers now (I am on windows and hence using .bat file)

```
start kafka-server-start.bat server0.properties
start kafka-server-start.bat server1.properties
start kafka-server-start.bat server2.properties
```

Multiple Brokers

- Start one producer

```
kafka-console-producer.bat --broker-list
"localhost:9090,localhost:9091,localhost:9092" --topic NEWS
```

- Start a couple of consumers

```
kafka-console-consumer.bat --topic NEWS --zookeeper localhost:2181
```

```
kafka-console-consumer.bat --topic NEWS --zookeeper localhost:2181
```

Consumer Groups

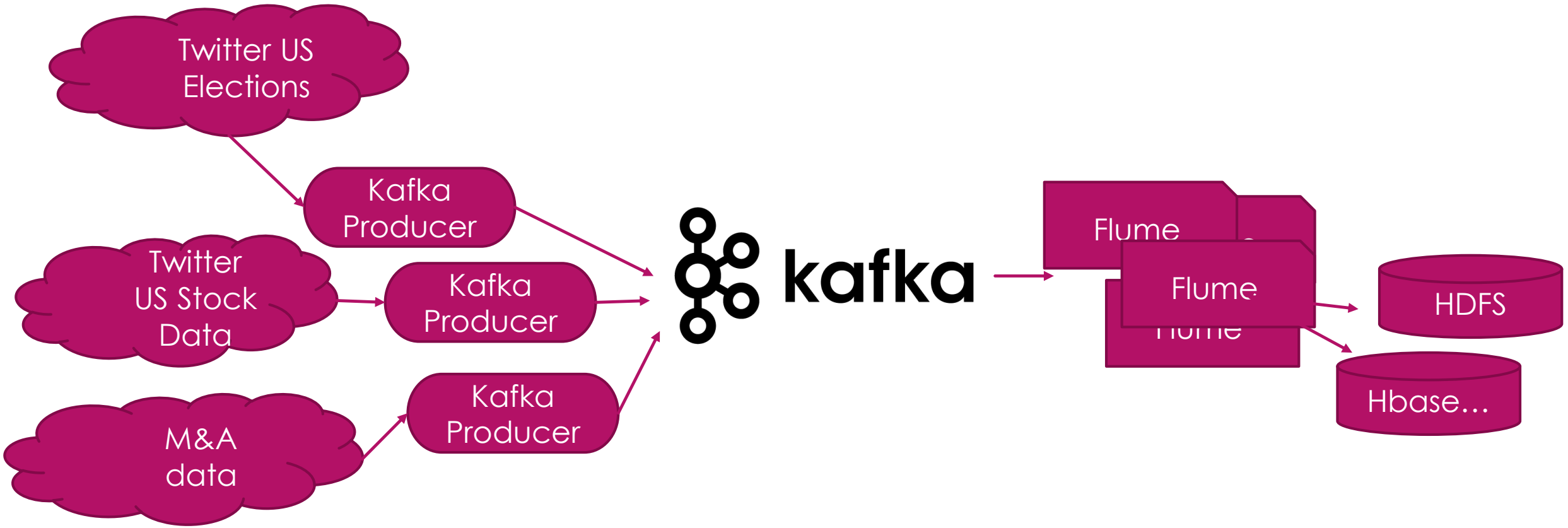
- Consumers can be segregated into groups using a group id, this allows for partitioning the messages for scale.
- Create a file named group.properties with following content

  group.id=g1
- Start a set of consumers with the new group configuration (leave the ones running earlier as is)
  - kafka-console-consumer.bat --topic NEWS --zookeeper localhost:2181 --consumer.config group.properties
  - kafka-console-consumer.bat --topic NEWS --zookeeper localhost:2181 --consumer.config group.properties
- Try sending some messages to our NEWS topic and observe the consumers.

Some code

- Writing a Producer
- Writing a Consumer

A Use case, Investment Portfolio App



A Sample flume cfg for Kafka Source

```
agent1.sources = kfksrc
agent1.sinks = logsink
agent1.channels = c1

agent1.sources.kfksrc.type = org.apache.flume.source.kafka.KafkaSource
agent1.sources.kfksrc.zookeeperConnect = localhost:2181
agent1.sources.kfksrc.topic = NEWS
agent1.sources.kfksrc.groupId = grp1_news
agent1.sources.kfksrc.kafka.consumer.timeout.ms = 100

agent1.sinks.logsink.type = logger

agent1.channels.c1.type = file

agent1.sources.kfksrc.channels = c1
agent1.sinks.logsink.channel = c1
```

Overlaps with Flume

- Kafka is very much a general-purpose system. You can have many producers and many consumers sharing multiple topics. In contrast, Flume is a special-purpose tool designed to send data to HDFS and HBase.
- Flume has many built-in sources and sinks which can be used straight out of the box. Kafka, however, requires coding.
- Flume can process data in-flight using interceptors. These can be very useful for data masking or filtering. Kafka requires an external stream processing system for that.
- Flume does not replicate events. As a result, even when using the reliable file channel, if a node with Flume agent crashes, you will lose access to the events in the channel until you recover the disks. Use Kafka if you need an ingest pipeline with very high availability.
- Flume and Kafka can work quite well together. If an application requires streaming data from Kafka to Hadoop, using a Flume agent with Kafka source to read the data makes sense: You don't have to implement your own consumer, you get all the benefits of Flume's integration with HDFS and HBase