Apache Kafka

AI Engineering - Recitation 1

Outline

Introduction & Preferences

- Recitations would mainly cover-
  - Software engineering concepts
  - Best practices
  - Collaboration strategies
  - Technologies
- Discussions would largely be helpful for the assignments and group project
- Designed to be around 40-50 minutes long
  - TA office hours for the remainder of the session

Introduction & Preferences

- What would you prefer?
  - More demos vs more hands-on activities
  - General discussions about broad concepts vs in-depth technical discussions

Synchronous vs Asynchronous

- Modern applications are usually distributed (multiple services operating on multiple systems)
    - How to communicate efficiently?
- Synchronous - sender and receiver both need to be active at the same time
    - Example: Rest API, RPC
- Asynchronous - sender and receiver need not be active at the same time
    - Example: Message queues, Kafka
- Choosing one depends on the type of communication
    - Synchronous: Point-to-point, on-demand messages
    - Asynchronous: Stream data, Pub/Sub

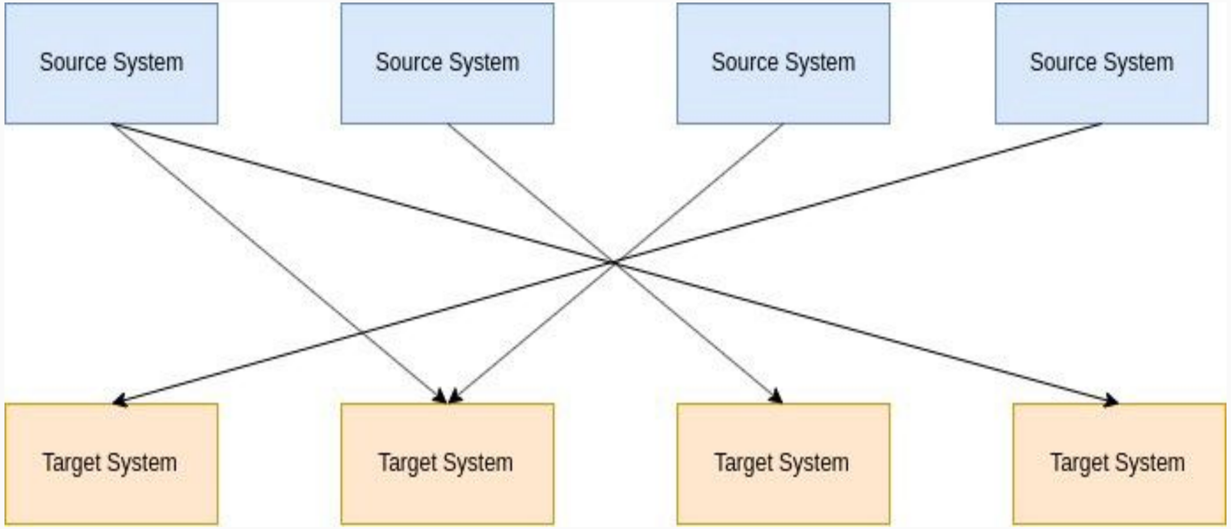
Why use Kafka?

- Message Queues are simple to use
  - Publish messages to one end and consume from another
  - Hold a large number of messages
  - Messages are always in order
- Message Queues lack some useful features
  - Cannot have multiple publishers and consumers on same queue
  - Scaling is a pain, need to practically setup entire queues again
  - Queue goes down, and all messages are lost

Why use Kafka?

- Kafka fixes many of the issues
    - Easy to scale
    - Resilient to failures, as messages are replicated
    - Allows for a Publisher/Subscriber model
    - Highly configurable as required
- Kafka is becoming a de facto standard for streaming data processing
    - Thanks to its versatility

Why use Kafka?

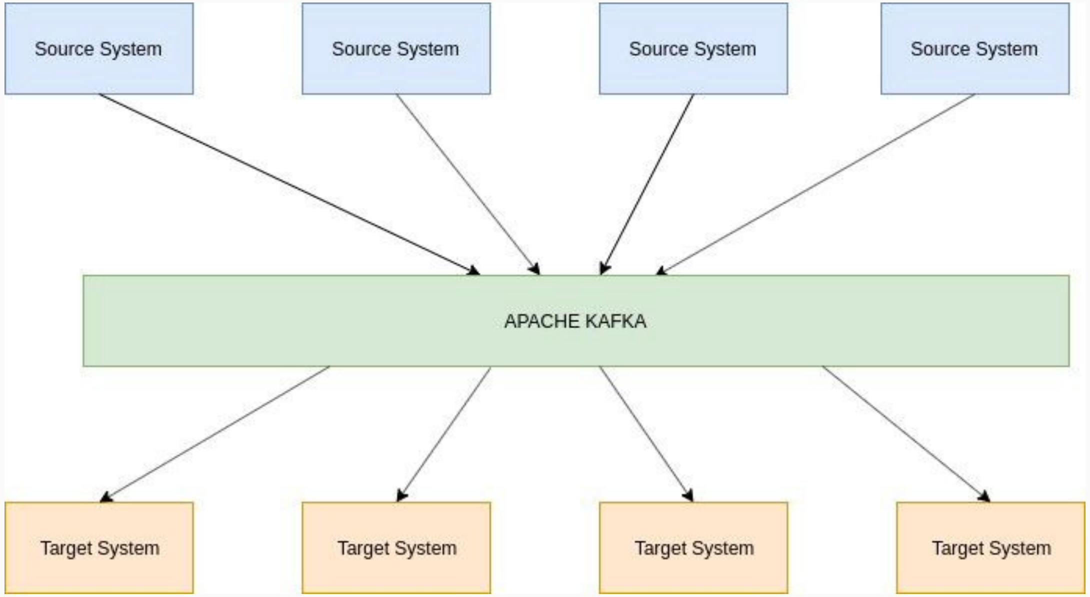


System with point-to-point communication.

Not scalable, every link needs to be established explicitly as source and targets increase.

Source: Getting Started With Apache Kafka

Why use Kafka?



Source and target systems now talk to Kafka.

Very scalable, Kafka handles all incoming and outgoing communication.

Systems do not need to know about each other to function.

Source: [Getting Started With Apache Kafka]

Why use Kafka?

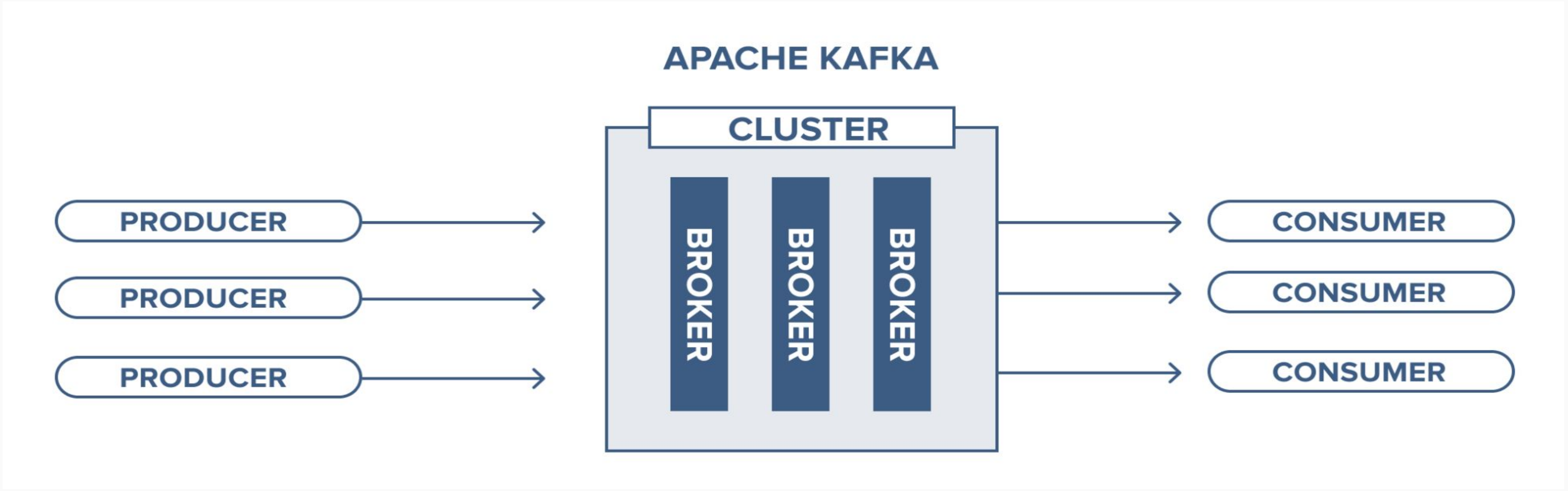
- Can you think of any disadvantages to this design?

- Follow up - Think about how each of these techniques complement each other
  - Real-world systems are seldom homogenous

Activity - Design decisions

- You have a system that accepts an audio clip as an input and transcribes it into text
  - How about video?
- You work for Uber, and you want to keep track of your drivers' location
- You are designing Amazon's recommendation system, and you want to gather users' preferences from their online activities.

Parts of a Kafka System

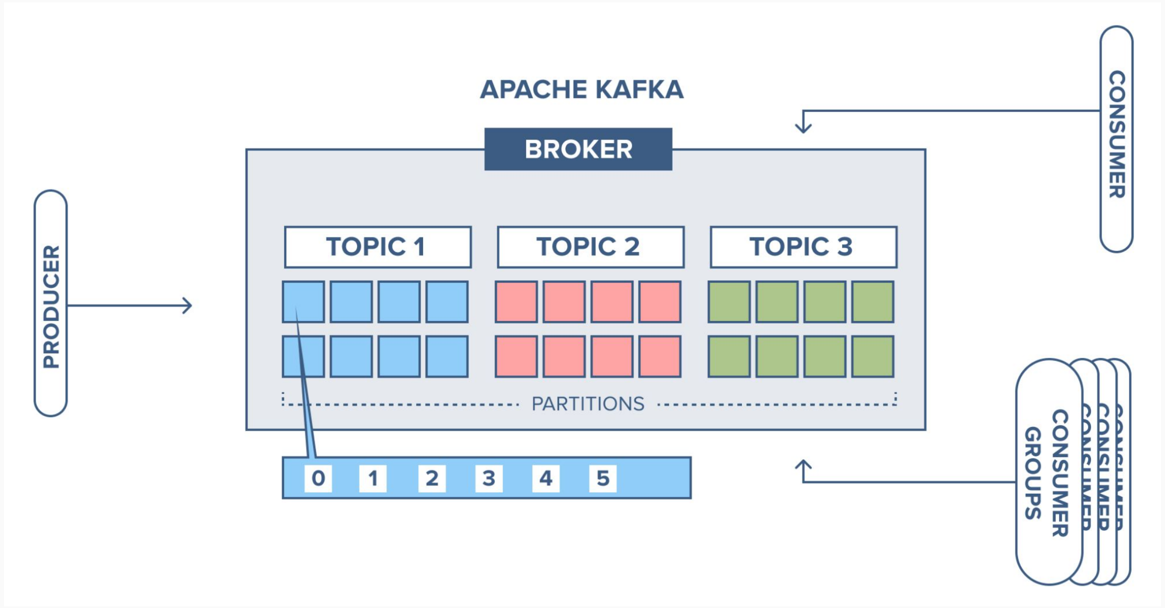


Source: [What is Apache Kafka?]

Parts of a Kafka System

- Producer: Applications or processes that produces messages
- Consumer: Applications or processes that consume messages
- Broker: A server that manages storage of messages
  - Multiple brokers come together to form a cluster
- Zookeeper: Service to manage brokers and monitor clusters

Topics, Partitions and Offsets in Kafka



Source: What is Apache Kafka?

Topics, Partitions and Offsets in Kafka

- Topic: How Kafka messages are stored and organized
    - Producers publish to topics, and consumers subscribe to topics
    - Single producer can publish to multiple topics; single consumer can subscribe to multiple topics

- Partitions: An ordered split of a topic
    - Done for increasing throughput via parallelism
    - Message ordering is only preserved within a partition
    - Messages within a partition are immutable

Topics, Partitions and Offsets in Kafka

- Offset: An identifier associated with a message within a partition
  - Used by consumer to read messages in any order
  - Typically, it would be linearly advanced

- Event: A message in Kafka is also referred to as an event
  - Events are nothing but byte arrays
  - Kafka inherently supports messages of any format required

SSH Tunneling

- Why SSH tunneling?
    - VMs on cloud are always vulnerable. Ports should be kept closed.
    - Using SSH, establish a secure tunnel to connect service running on a port on VM to your local system
    - Generic way to treat remotely hosted services as if they're running locally

- How to set up a tunnel?

```
ssh -L <local_port>:localhost:<remote_port> <user>@<remote_server> -NTf

## Understanding the options used left as an exercise
```

Demo - Kafkacat utility and kafka-python

- Kafkacat: CLI tool for interacting with Kafka broker
  - Produce and consume events
  - Manage topics
- kafka-python: Python library for Kafka

- In the demo, we would be-
  - Opening an SSH tunnel to Kafka broker
  - Example usage of Kafkacat and kafka-python