# Project Milestone 2: Data Ingestion Software – Kafka Clusters

*Raveenth Maheswaran 100704540*

## Concepts of Event-Driven Architecture

**What is EDA? What are its advantages and disadvantages?**

- Event-driven architecture (EDA) is a well-thought-out design that utilizes events to prompt its associated service/task for its intended purpose
- Events can be defined as a 'change of state'
  - For instance, when a user turns on a light, the light's state changed from off to on
  - In the case of event-driven architecture, real-time data triggers an event where the data can be processed however it may be
- Advantages of EDA
  - Fault tolerance
    - A fault from a service does not affect the entirety of the system due to decoupling
  - Highly scalable
    - Since EDA uses a serverless stack, it can dynamically handle requests by increasing the concurrency of executing functions
  - Better user experience
    - Implementing real-time data to a project/application is a lot easier using EDA, as it provides the associated functionality to do so
- Disadvantages of EDA
  - Troubleshooting inefficiencies
    - Monitoring producers and consumers is tough since there are multiple of each
    - Would need to use third-party software tools to monitor the data flow
  - Over-engineered design
    - In certain cases, a simple API call between services can get the job done
    - Implementing EDA into a system requires a lot of setup and design choices, resulting in higher costs and longer deadlines

**In Kafka, what's meant by cluster, broker, topic, replica, partition, zookeeper, controller, leader, consumer, producer, and consumer group?**

- Cluster
  - A Kafka cluster consists of a group of brokers
  - Also includes a zookeeper
- Broker
  - Kafka servers, aka brokers, are used to retrieve messages for consumers by their topic, partition, and offset

    - Multiple brokers can communicate with each other via zookeeper
- Topic
    - Kafka topic is a named category, used to keep messages organized
    - Can be used in multiple producers and consumers
    - Typically named after its intended use
- Replica
    - Kafka replica is used to duplicate data and save it across multiple brokers
- Partition
    - Kafka partitioning takes the messages from a topic and break it down into multiple logs, to be stored in different locations
- Zookeeper
    - Kafka zookeeper provides synchronization between the multiple brokers in the same Kafka cluster
- Controller
    - Kafka controllers are responsible for the state of the partitions and replicas within the broker
- Leader
    - Kafka leader is the main priority of a partition among the followers in other partitions
    - Leader is responsible for any changes
- Consumer
    - Kafka consumer is a subscriber that subscribes to a topic to listen to messages produced from the producer/publisher
- Producer
    - Kafka producer is a publisher where it produces messages to a topic
- Consumer Group
    - Kafka consumer group is a group of consumers where they cooperate among each other, via a group ID

## Local Machine Kafka Installation + Basic Kafka Network Configuration

Video Demonstration:

https://drive.google.com/file/d/1HG8bVK3W84dO1TqV7ZCOSSLRsC0puiEb/view?usp=sharing

## Kafka NodeJS Scripts

Video Demonstration:

https://drive.google.com/file/d/1oNhnacUY_CoUIDyULohSakU0ivIKLvbJ/view?usp=sharing

## Kafka Python Scripts

Video Demonstration:

https://drive.google.com/file/d/1FV0ska2UVed0ErrVYQAA4eKwdVXZu0JU/view?usp=sharing

## Persistent Data in YAML File

Zookeeper:

```
"Image": "confluentinc/cp-zookeeper"
"Volumes": {
    "/etc/zookeeper/secrets": {},
    "/var/lib/zookeeper/data": {},
    "/var/lib/zookeeper/log": {}
},
```

```
zookeeper:
  image: confluentinc/cp-zookeeper
  hostname: zookeeper
  container_name: zookeeper
  networks:
    - kafka_Network
  ports:
    - 2181:2181
  environment:
    ZOOKEEPER_CLIENT_PORT: 2181
    ZOOKEEPER_TICK_TIME: 2000
  volumes:
    - /etc/zookeeper/secrets
    - /var/lib/zookeeper/data
    - /var/lib/zookeeper/log
```

Brokers:

```
],
"Image": "confluentinc/cp-kafka",
"Volumes": {
    "/etc/kafka/secrets": {},
    "/var/lib/kafka/data": {}
},
"WorkingDir": "/home/appuser",
```

```
broker1:
  image: confluentinc/cp-kafka
  hostname: broker1
  container_name: broker1
  networks:
    - kafka_Network
  depends_on:
    - zookeeper
  ports:
    - 9093:9093
  expose:
    - 9093
  environment:
    KAFKA_BROKER_ID: 1
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:P
    KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
    KAFKA_ADVERTISED_LISTENERS: INTERNAL://broker1:9
    KAFKA_DEFAULT_REPLICATION_FACTOR: 3
    KAFKA_NUM_PARTITIONS: 3
  volumes:
    - /etc/kafka/secrets
    - /var/lib/kafka/data
```

```
broker2:
  image: confluentinc/cp-kafka
  hostname: broker2
  container_name: broker2
  networks:
    - kafka_Network
  depends_on:
    - zookeeper
  ports:
    - 9094:9094
  expose:
    - 9094
  environment:
    KAFKA_BROKER_ID: 2
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:P
    KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
    KAFKA_ADVERTISED_LISTENERS: INTERNAL://broker2:9
  volumes:
    - /etc/kafka/secrets
    - /var/lib/kafka/data
```

```
broker3:
  image: confluentinc/cp-kafka
  hostname: broker3
  container_name: broker3
  networks:
    - kafka_Network
  depends_on:
    - zookeeper
  ports:
    - 9095:9095
  expose:
    - 9095
  environment:
    KAFKA_BROKER_ID: 3
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:P
    KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
    KAFKA_ADVERTISED_LISTENERS: INTERNAL://broker3:9
  volumes:
    - /etc/kafka/secrets
    - /var/lib/kafka/data
```

| Name | Date modified | Type | Size |
|---|---|---|---|
| broker1 | 2/15/2022 9:52 AM | File folder | |
| broker2 | 2/15/2022 9:52 AM | File folder | |
| broker3 | 2/15/2022 9:52 AM | File folder | |
| Confluent Kafka Python | 2/15/2022 12:10 AM | File folder | |
| Kafka NodeJS | 2/13/2022 1:39 PM | File folder | |
| Kafka Python | 2/13/2022 9:46 PM | File folder | |
| zoo | 2/15/2022 9:52 AM | File folder | |
| Data Ingestion Software - Kafka Clusters | 2/15/2022 10:52 AM | Microsoft Word D... | 161 KB |
| docker-compose | 2/15/2022 10:05 AM | YML File | 3 KB |

In order to implement persistent data to a broker/multiple brokers and zookeepers, we can add the volume path to 'volumes' in the YAML file so that the data can be stored accordingly.

## Confluent Kafka CLI and Python Scripts

https://drive.google.com/file/d/11BaDij7sXLog0QER9nthbKFz7rHTlgtj/view?usp=sharing