Understanding Kafka Partitions – Lab: 2

Overview of Partitions

What Are Partitions?

- A partition is a sub-division of a Kafka topic.
- Think of a topic as a bookshelf, and partitions are separate shelves. Each partition stores a portion of the data for that topic.

How Partitions Work in Kafka?

1. Producers and Partitions:
- Producers can send messages to specific partitions by directly specifying the partition.

2. Consumers and Partitions:
- Consumers can either read from all partitions of a topic or subscribe to specific partitions only.
- This allows consumers to focus on specific types of data or work in parallel to process large volumes efficiently.

Why Are Partitions Important?

1. Scalability:
- Imagine a single partition is like a cashier at a busy store. If there are too many customers (messages), the cashier gets overwhelmed.
- By dividing a topic into multiple partitions, Kafka acts like a store with multiple cashiers, distributing the workload evenly. (It is a default application of Kafka)

2. Fault Tolerance:
- Partitions also have replicas which can replicate data on other Kafka servers (brokers). If one broker fails, another broker takes overusing the replica, ensuring no data is lost.

3. Efficient Data Access:
- Kafka lets consumers read from specific partitions. This is useful when certain consumers need specific types of data or when data processing needs to be divided among multiple consumers.

Examples of Its Applications

1. E-commerce Order Processing:
- A topic orders might have 3 partitions:
  - Partition 0: Orders from North America.
  - Partition 1: Orders from Europe.
  - Partition 2: Orders from Asia.

  Each region's orders are stored in their respective partitions, making it easy to process them in parallel.

2. Banking Transactions:
- A topic transaction can be partitioned by transaction type.
  - Partition 0: Deposits.
  - Partition 1: Withdrawals.
  - Partition 2: Transfers.

  This separation allows the bank to handle each type of transaction with dedicated processing systems.

Lab Steps

Open CLI and navigate into the kafka directory using "cd" command.

Step 1: Starting Kafka

1. Start Zookeeper:

bin/zookeeper-server-start.sh config/zookeeper.properties

2. Start Kafka Broker:
   bin/kafka-server-start.sh config/server.properties
3. List Topics:
   Verify Kafka is running by listing existing topics:
   bin/kafka-topics.sh --list --bootstrap-server localhost:9092

Step 2: Creating the Topic

Create a topic with 2 partitions for this lab using the following command:

bin/kafka-topics.sh --create --topic partition_topic --partitions 2 --bootstrap-server localhost:9092

Use the following command to verify the topic creation:

bin/kafka-topics.sh --describe --topic partition_topic --bootstrap-server localhost:9092

Step 3: Preparing the Producer and Consumer

- Producer: Use the provided code PartitionProducer.py to send messages to a specific partition.
- Consumer: Use the provided code PartitionConsumer.py to read messages from specific partitions.

Ensure both files are in the same directory and properly configured.

Step 4: Running the Producer

1. Navigate to the directory containing PartitionProducer.py.
2. Execute the producer code:
   python3 PartitionProducer.py
3. When prompted, specify the target partition (0 or 1) and provide the data for that partition.

Step 5: Running the Consumer

1. Open a new terminal tab.
2. Navigate to the directory containing PartitionConsumer.py.
3. Execute the consumer code:
   python3 PartitionConsumer.py
4. The consumer will read messages from both partitions by default. Modify the script if required to read from only one partition.

Expected Output

- The producer will send messages to the specified partition and confirm delivery.
- The consumer will display the messages it receives along with partition and offset details.