A Practical Guide to Apache Kafka CLI Commands

This guide provides a hands-on overview of essential Kafka Command Line Interface (CLI) tools. The examples are tailored for a 3-node Kafka cluster and assume kafka-199:9092 is your bootstrap server.

Prerequisites

- A running Apache Kafka cluster (3 nodes recommended).
- Access to the kafka/bin directory on one of the cluster nodes.
- The sudo -u kafka prefix is used for all commands to ensure they run with the correct user permissions.

1. Creating a Topic

A topic is a category or feed name to which records are published. This command creates a topic named test-topic with 3 partitions and a replication factor of 3.

Command:

```
sudo -u kafka /usr/local/kafka/bin/kafka-topics.sh --create \
  --topic test-topic \
  --partitions 3 \
  --replication-factor 3 \
  --bootstrap-server kafka-199:9092
```

- --create: The action to perform.
- --topic: The name of the topic.
- --partitions: The number of partitions. For optimal performance in a 3-node cluster, 3 is a good starting point.
- --replication-factor: The number of replicas for each partition. A replication factor of 3 means each partition will have two additional replicas, providing fault tolerance.
- --bootstrap-server: The address of at least one broker to connect to.

2. Producing Messages with the Console Producer

The console producer is a simple tool to send messages to a topic from the command line.

Command:

```
sudo -u kafka /usr/local/kafka/bin/kafka-console-producer.sh \
```

```
  --topic test-topic \
  --bootstrap-server kafka-199:9092
```

After running the command, you can type messages and press Enter. Each line will be sent as a separate message to the test-topic.

```
>Hello from the producer
>This is my second message
>One more message
```

3. Producing Messages with a Key

A key is crucial for ensuring that messages with the same key always go to the same partition. This is essential for maintaining message order and state.

To produce messages with a key, you must use --property parse.key=true and specify a --property key.separator. The format is key:value.

Command:

```
sudo -u kafka /usr/local/kafka/bin/kafka-console-producer.sh \
  --topic test-topic \
  --bootstrap-server kafka-199:9092 \
  --property parse.key=true \
  --property key.separator=:
```

Example:

```
>user-A:login successful
>user-B:account created
>user-A:logout
>user-C:viewed profile
```

- user-A:login successful and user-A:logout will be guaranteed to land on the same partition because they share the key user-A.

4. Distribution of Messages Across Partitions

Kafka uses a hash-based partitioning strategy. When a key is provided, a hash of the

key determines which partition the message is sent to. If no key is provided, messages are sent to partitions in a round-robin fashion.

Describe the topic to see partition leaders:

```
sudo -u kafka /usr/local/kafka/bin/kafka-topics.sh --describe \
  --topic test-topic \
  --bootstrap-server kafka-199:9092
```

Expected Output:

```
Topic: test-topic    TopicId: ...    PartitionCount: 3    ReplicationFactor: 3Configs:
        Partition: 0    Leader: 1    Replicas: 1,2,3    Isr: 1,2,3
        Partition: 1    Leader: 2    Replicas: 2,3,1    Isr: 2,3,1
        Partition: 2    Leader: 3    Replicas: 3,1,2    Isr: 3,1,2
```

This shows that each partition has a leader, replicas, and an In-Sync Replica (ISR) set.

5. Consuming Messages with the Console Consumer

The console consumer is a simple tool to read messages from a topic.

Consuming without a Key

This is the default behavior.

Command:

```
sudo -u kafka /usr/local/kafka/bin/kafka-console-consumer.sh \
  --topic test-topic \
  --bootstrap-server kafka-199:9092 \
  --from-beginning
```

- --from-beginning: Reads all messages from the very beginning of the topic. If this flag is omitted, it will only consume new messages.

Consuming with a Key

To see the key associated with each message, you need to add the --property print.key=true and specify the deserializers.

Command:

```
sudo -u kafka /usr/local/kafka/bin/kafka-console-consumer.sh \
  --topic test-topic \
  --bootstrap-server kafka-199:9092 \
  --from-beginning \
  --property print.key=true \
  --property
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

Example Output:

```
user-A      login successful
user-A      logout
user-B      account created
user-C      viewed profile
```

6. Consumer Groups

A consumer group is a set of consumers that cooperate to consume data from one or more topics. Each partition is consumed by a single consumer in the group, ensuring that messages are not processed multiple times.

Starting a Consumer Group

Open a terminal and start the first consumer in a new consumer group named my-consumer-group.

Consumer 1 (Terminal 1):

```
sudo -u kafka /usr/local/kafka/bin/kafka-console-consumer.sh \
  --topic test-topic \
  --bootstrap-server kafka-199:9092 \
  --group my-consumer-group \
  --from-beginning
```

You will see this consumer start receiving messages.

Adding a Second Consumer

Open a **second terminal** and start a second consumer with the **same group ID**.

Consumer 2 (Terminal 2):

```
sudo -u kafka /usr/local/kafka/bin/kafka-console-consumer.sh \
  --topic test-topic \
  --bootstrap-server kafka-199:9092 \
  --group my-consumer-group \
  --from-beginning
```

- **Observation:** The two consumers will automatically rebalance. Each consumer will now be assigned a subset of the topic's partitions, and you will see messages appearing across both terminals, with no message being duplicated.

7. Consumer Offset

The consumer offset is a pointer to the last message a consumer group has successfully processed for a specific partition. Kafka stores this offset, allowing the consumer group to resume from where it left off.

Viewing Consumer Group Offsets

You can use the kafka-consumer-groups.sh tool to inspect the state of a consumer group.

Command:

```
sudo -u kafka /usr/local/kafka/bin/kafka-consumer-groups.sh \
  --bootstrap-server kafka-199:9092 \
  --describe \
  --group my-consumer-group
```

Expected Output:

```
GROUP           TOPIC         PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG
CONSUMER-ID                           HOST          CLIENT-ID
my-consumer-group  test-topic    0          2               2               0
console-consumer-19760-496e... /172.17.0.3    console-consumer
my-consumer-group  test-topic    1          1               1               0
console-consumer-57999-5211... /172.17.0.4    console-consumer
my-consumer-group  test-topic    2          3               3               0
```

console-consumer-19760-496e... /172.17.0.3 console-consumer

- CURRENT-OFFSET: The last message offset that the consumer group has committed.
- LOG-END-OFFSET: The offset of the last message in the partition.
- LAG: The number of messages the consumer group has not yet processed (LOG-END-OFFSET - CURRENT-OFFSET). A lag of 0 means the consumer group is caught up.
- CONSUMER-ID: The unique ID of the consumer instance currently assigned to the partition.

This command is invaluable for monitoring the health and progress of your Kafka consumer applications.