**Apache Kafka with KRaft (Kafka 4.0+)**

KRaft (Kafka Raft Metadata mode) replaces ZooKeeper in Kafka 4.0 and above. Here are the key

reasons why:

1. **Simplified Architecture:**

- No external ZooKeeper dependency.

- Kafka handles its own metadata natively.

2. **Faster and More Reliable Controller Failover:**

- Raft-based leader election is faster and more robust than ZooKeeper-based.

3. **Consistent Metadata:**

- All metadata changes are replicated via Raft.

- Strong consistency guarantees.

4. **Better Cloud and Kubernetes Integration:**

- Suits dynamic, containerized environments.

5. **Future-Proof:**

- ZooKeeper is removed starting Kafka 4.0.

- KRaft is the only supported metadata system.

**Kafka Storage Utility and UUID Usage**

**Understanding kafka-storage.bat and UUID:**

- kafka-storage.bat is a command-line tool to manage Kafka storage directories for brokers in KRaft

mode.

- It initializes and formats the storage directory with a unique cluster UUID.

- This UUID identifies the metadata quorum and ensures all brokers belong to the same Kraft cluster.

**Steps:**

1. Generate a unique UUID for your cluster:

sh kafka-storage.sh random-uuid

2. Format the broker's storage directory with the generated UUID:

sh kafka-storage.sh format --standalone -t <UUID> ../config/kraft/server.properties

- In multi-broker clusters, use the same UUID for all brokers.

- Once formatted, the broker's storage directory is ready to store metadata and logs.

**Single Kafka Broker Setup**

Steps to Start a Single Kafka Broker (Kafka 4.0+ with KRaft on Windows):

1. Generate a UUID for your KRaft cluster:

sh kafka-storage.sh random-uuid

2. Format the storage directory using the UUID:

sh kafka-storage.sh format --standalone -t <UUID> -c ../config/kraft/server.properties

3. Start the Kafka server:

sh kafka-server-start.sh ../config/kraft/server.properties

**Two-Broker Kafka Cluster with Kraft**

Steps to Set Up a Two-Broker Kafka Cluster with KRaft:

1. Duplicate the configuration file:

- Create config/broker1.properties and config/broker2.properties

2. Configure broker1.properties:

node.id=1

process.roles=broker,controller

controller.quorum.voters=1@localhost:9093,2@localhost:9094

listeners=PLAINTEXT://localhost:9092,CONTROLLER://localhost:9093

inter.broker.listener.name=PLAINTEXT

log.dirs=C:/kafka/logs-broker1

3. Configure broker2.properties:

node.id=2

process.roles=broker,controller

controller.quorum.voters=1@localhost:9093,2@localhost:9094

listeners=PLAINTEXT://localhost:9095,CONTROLLER://localhost:9094

inter.broker.listener.name=PLAINTEXT

log.dirs=C:/kafka/logs-broker2

4. Generate one UUID to use for both brokers:

sh kafka-storage.sh random-uuid

5. Format both brokers with the same UUID:

sh kafka-storage.sh format -t <UUID> -c ../config/kraft/broker1.properties

sh kafka-storage.sh format -t <UUID> -c ../config/kraft/broker2.properties

6. Start each broker in separate terminals:

sh kafka-server.sh -config ../config/kraft/broker1.properties

sh kafka-server.sh -config ../config/kraft/broker2.properties

**Topic, Producer and Consumer Commands**

Kafka Topic, Producer and Consumer Commands:

1. Create a topic:

sh kafka-topics.sh --create --topic test-topic --bootstrap-server localhost:9092 --partitions 1

--replication-factor 2

2. List topics:

sh kafka-topics.sh --list --bootstrap-server localhost:9092

3. Describe a topic:

sh kafka-topics.sh --describe --topic test-topic --bootstrap-server localhost:9092

4. Start a producer to send messages:

sh kafka-console-producer.sh --topic test-topic --bootstrap-server localhost:9092

5. Start a consumer to read messages:

sh kafka-console-consumer.sh --topic test-topic --from-beginning --bootstrap-server localhost:9092