Documentation on Modifying __cluster_metadata in Kafka KRaft Mode

I want to modify __cluster_metadata; can you describe the structure?

Kafka in KRaft (Kafka Raft) mode uses a dedicated internal topic named __cluster_metadata to manage critical cluster metadata.

This topic replaces the role traditionally held by Zookeeper in managing Kafka cluster metadata, configuration, and state.

Below is a detailed description of the structure and key components of the __cluster_metadata directory.

---

1. Introduction to __cluster_metadata Structure

In KRaft mode, Kafka's internal metadata (such as topic configurations, broker registrations, partition assignments, ACLs, and controller state) is managed by the Raft consensus protocol and stored in the __cluster_metadata directory.

This directory functions similarly to a topic partition directory but is specialized for storing the Raft log, which Kafka controllers use to synchronize and replicate the cluster's metadata.

Typical Structure of __cluster_metadata Directory

The __cluster_metadata directory contains log segments, index files, checkpoint files, and snapshots, all of which play roles in managing the metadata and ensuring consistency.

---

2. Key Components in __cluster_metadata

Each file in the __cluster_metadata directory has a specific purpose, as detailed below:

- **Log Segments (.log)**:

Log segments, such as 00000000000000000000.log, 00000000000000000001.log, etc., store records of metadata changes (Raft records).

Each log file stores metadata updates in sequence, including topic creation, broker registration, configuration updates, etc.

- **Index Files (.index and .timeindex)**:

The .index file helps locate entries within a log segment, while the .timeindex file enables time-based lookups. These files improve the efficiency of accessing specific entries within the metadata log.

- **Leader-Epoch Checkpoint (leader-epoch-checkpoint)**:

This file maps each epoch to an offset in the log where a leader election occurred. It helps track changes in leadership for the metadata quorum.

- **Checkpoint (checkpoint)**:

The checkpoint file marks the most recent committed offset for the Raft quorum's metadata log, assisting Raft followers in catching up on metadata changes.

- **Snapshot Directory (snapshot/)**:

  This folder contains snapshots of the complete metadata state at specific points, allowing for efficient recovery by reducing the need to replay all logs.

---

3. Example Data Entries in Log Segments

Each log segment in the __cluster_metadata folder may contain records representing different metadata actions, such as:

- **Topic Creation (TopicRecord)**:
  - Fields include topic name, number of partitions, replication factor, and configurations.
  - Example:

```
    {
      "record_type": "TopicRecord",
      "topic_name": "example_topic",
      "num_partitions": 3,
      "replication_factor": 2
    }
```

- **Partition Assignment (PartitionRecord)**:

  - Details about broker assignments for each partition.

  - Example:

```
    {
      "record_type": "PartitionRecord",
      "topic_name": "example_topic",
      "partition_id": 0,
      "replica_nodes": [1, 2]
    }
```

- **Configuration Update (ConfigRecord)**:

  - Changes to topic settings or broker configurations.

  - Example:

```
    {
      "record_type": "ConfigRecord",
      "config_key": "retention.ms",
      "config_value": "86400000"
    }
```

- **Broker Registration (BrokerRegistrationRecord)**:

  - Registers or deregisters brokers in the cluster.

  - Example:

```
    {
      "record_type": "BrokerRegistrationRecord",
      "broker_id": 1,
      "endpoints": [{"listener_name": "PLAINTEXT", "host": "broker1", "port": 9092}]
    }
```

---

4. Recommendations for Modifying __cluster_metadata

Modifying __cluster_metadata directly is generally not recommended, as it can lead to inconsistencies or data corruption in the cluster metadata. Instead, use Kafka's Admin API or kafka-configs.sh and kafka-topics.sh for safe modifications.

Summary

The __cluster_metadata directory structure is critical for the operation of Kafka in KRaft mode, as it stores all cluster metadata in a consistent and durable way using the Raft consensus mechanism. This structure includes log segments, checkpoints, snapshots, and indexes to manage and replicate cluster metadata effectively.

This documentation provides an overview of the structure and key components within the __cluster_metadata directory, which Kafka uses to maintain essential cluster state data in KRaft mode.