KIP-835: Monitor KRaft Controller Quorum Health

Status

Current state: Approved

Discussion thread: *here*

JIRA: *KAFKA-13883*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

With KRaft, Kafka added a new controller quorum to the cluster. These controllers needs to be able to commit records for Kafka to be available. One way to measure availability, is by periodically causing the high-watermark and the last committed offset to increase. Monitoring services can compare that these last committed offsets are advancing. They can also use these metrics to check that all of the brokers and controllers are relatively within each other's offset.

Public Interfaces

Cluster Metadata Records

NoOpRecord

Add a new record to periodically advancement of the LEO and high-watermark. Controller or broker state will not change when applying this record. This record will not be included in the cluster metadata snapshot.

```
{
  “apiKey”: TBD,
  “type”: “metadata”,
  “name”: “NoOpRecord”,
  “validVersions”: “0”,
  “flexibleVersions”: “0+”,
  “fields”: []
}
```

Metrics

Controller

1. `kafka.controller:type=KafkaController,name=MetadataLastAppliedRecordOffset`
   Reports the offset of the last record consumed by the controller. For the active controller this may include uncommitted records. For the inactive controller this always includes committed records.
2. `kafka.controller:type=KafkaController,name=MetadataLastCommittedRecordOffset`
   The active controller will report the offset of that last committed offset it consumed. Inactive controllers will always report the same value in MetadataLastAppliedRecordOffset.

3. `kafka.controller:type=KafkaController,name=MetadataLastAppliedRecordTimestamp`
   Reports the append time of the last applied record batch.
4. `kafka.controller:type=KafkaController,name=MetadataLastAppliedRecordLagMs`
   Reports the difference between the local time and the append time of the last applied record batch.

Broker

1. `kafka.server:type=broker-metadata-metrics,name=last-applied-record-offset`
   Reports the offset of the last record consumed by the broker.
2. `kafka.server:type=broker-metadata-metrics,name=last-applied-record-timestamp`
3. `kafka.server:type=broker-metadata-metrics,name=last-applied-record-lag-ms`
4. `kafka.server:type=broker-metadata-metrics,name=pending-record-processing-time-us-avg`
   Reports the average amount of time it took for the broker to process all pending committed records when there are pending records in the cluster metadata partition. The time unit for this metric is microseconds.
5. `kafka.server:type=broker-metadata-metrics,name=pending-record-processing-time-us-max`
   Reports the maximum amount of time it took for the broker to process all pending committed records when there are pending records in the cluster metadata partition. The time unit for this metric is microseconds.
6. `kafka.server:type=broker-metadata-metrics,name=record-batch-size-byte-avg`
   Reports the average byte size of the record batches in the cluster metadata partition.
7. `kafka.server:type=broker-metadata-metrics,name=record-batch-size-byte-max`
   Reports the maximum byte size of the record batches in the cluster metadata partition.

Configuration

1. metadata.max.idle.interval.ms - The interval between writing NoOpRecord to the cluster metadata log. The default for this value is 500 ms.

Proposed Changes

Active Controller

The active controller will increase the LEO and high-watermark by periodically writing a no-op record (NoOpRecord) to the metadata log. The active controller will write this new record only if the IBP and metadata.version supports this feature. See the backward compatibility section for details.

The implementation will only append the NoOpRecord, if the LEO wasn't already advanced in the defined period.

Controllers and Brokers

In both the controller and broker, the metadata replaying code will be extended to ignore NoOpRecord.

Compatibility, Deprecation, and Migration Plan

The IBP and metadata.version will be bumped. This feature and record will only be produced if the active controller is at the expected version or greater.

Users must use the same software version of `DumpLogSegment` associated with the server node when reading the metadata cluster log segments.

Rejected Alternatives

Control Records

Instead of using the NoOpRecord metadata record. We could have added a control record in the KRaft layer. This solution has two problems.

1. Control records in KRaft are not exposed to the controller and broker KRaft listener. This means that those listeners will not update their last committed offset when those records get committed. This would make it difficult to make the last committed metrics represent what the broker and controllers observe.
2. Those records will not get snapshotted. It is possible for active controllers to never write to the metadata log if the user doesn't perform any admin or metadata operations. This means that KRaft will write these control records to the log without a mechanism for snapshotting (removing) those records.

Max Lag from the Active Controller

It is possible for the active controller to report the max lag from all of the brokers. The brokers send the last committed offset that they read to the active controller. The controller can compute the maximum of these values and report it as a metric.

This works for brokers but it doesn't work for controllers. The controllers don't send metadata heartbeat RPCs.