KIP-841: Fenced replicas should not be allowed to join the ISR in KRaft

Status

Current state: *Approved*

Discussion thread: *here*

JIRA: *here*

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

Motivation

When a follower catches up with the leader, the leader tries to add it back to the ISR. The AlterPartition API is used by the leader to persist the new ISR in the controller. Presently, the controller validates that the new ISR contains valid replicas of the partition but without taking their state into account - a leader could for instance add a fenced or shutting down replica to the ISR. This means that we always trust that the leader will do the right thing. We believe that we should be more defensive and ensure that fenced and shutting replicas are not allowed to join the ISR in KRaft.

Proposed Changes

This KIP proposes enforcing the following two invariants:

1. a fenced or in-controlled-shutdown replica is not eligible to be in the ISR; and
2. a fenced or in-controlled-shutdown replica is not eligible to become leader.

The controlled shutdown state is not persisted to the metadata log at the moment. The information is only available as a soft state in the BrokerHeartbeartManager. This means that the information is not available to the leaders nor available after a fail-over of the controller. This KIP proposes persisting this state to the metadata log when a broker requests a controlled shutdown. The broker will leave this state when it registers itself with a new incarnation id.

On the controller side, there are a few cases to consider. Firstly, the controller should only put active (not fenced nor in-controlled-shutdown) replicas to the ISR when creating partitions. Secondly, it should also ensure that only active replicas can be elected as leaders when creating partitions or when a new leader is elected (e.g. preferred leader election or unplanned leader election). Thirdly, the controller will reject any AlterPartition request containing an ineligible replica in the new ISR with the newly introduced INELIGIBLE_REPLICA error code.

On the leader side, the leader will only consider active replicas to be eligible to join the ISR. It will rely on the metadata cache to get this information via the metadata log. The goal is to avoid sending unnecessary AlterPartition requests to the controller. As the metadata cache is eventually consistent, the leader might try to add a replica - which was just removed by the controller - back to the ISR because it does not know that the replica's latest state yet. As explained earlier, the controller with reject the ISR expansion with a INELIGIBLE_REPLICA error code. The leader is expected to revert back its state to the last committed state - assuming that the state did not change in the mean time. When a broker is unfenced by the controller, the leader does nothing because subsequent fetch requests from the followers will try to get them back into the ISR if they are caught-up.

While changing the AlterPartition API, the KIP proposes introducing a new NEW_LEADER_ELECTED error code that will be used in the case where the AlterPartition call completes a reassignment and the completed reassignment no longer include the leader. At the moment, FENCED_LEADER_EPOCH error code is used but it is misleading for operators.

Public Interfaces

New Error Code

INELIGIBLE_REPLICA - The new ISR contains at least one ineligible replica.

NEW_LEADER_ELECTED - The AlterPartition request successfully updated the partition state but the leader has changed.

AlterPartition RPC

The version of the AlterPartition API is bumped to version 2.

Request

As we need to bump the version anyway, we propose to include the `TopicId` field to replace `TopicName` in the request.

```
{
  "apiKey": 56,
  "type": "request",
  "listeners": ["zkBroker", "controller"],
  "name": "AlterPartitionRequest",
  // Version 1 adds LeaderRecoveryState field (KIP-704).
  //
  // Version 2 adds TopicId field to replace TopicName field (KIP-841).
  "validVersions": "0-2",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "BrokerId", "type": "int32", "versions": "0+", "entityType": "brokerId",
      "about": "The ID of the requesting broker" },
    { "name": "BrokerEpoch", "type": "int64", "versions": "0+", "default": "-1",
      "about": "The epoch of the requesting broker" },
    { "name": "Topics", "type": "[]TopicData", "versions": "0+", "fields": [
      { "name": "TopicName", "type": "string", "versions": "0-1", "ignorable": true, "entityType": "topicName",
        "about": "The name of the topic to alter ISRs for" },
      // New Field Begin
      { "name": "TopicId", "type": "uuid", "versions": "2+", "ignorable": true,
        "about": "The ID of the topic to alter ISRs for" },
      // New Field End
      { "name": "Partitions", "type": "[]PartitionData", "versions": "0+", "fields": [
        { "name": "PartitionIndex", "type": "int32", "versions": "0+",
          "about": "The partition index" },
        { "name": "LeaderEpoch", "type": "int32", "versions": "0+",
          "about": "The leader epoch of this partition" },
        { "name": "NewIsr", "type": "[]int32", "versions": "0+", "entityType": "brokerId",
          "about": "The ISR for this partition" },
        { "name": "LeaderRecoveryState", "type": "int8", "versions": "1+", "default": "0",
          "about": "1 if the partition is recovering from an unclean leader election; 0 otherwise." },
        { "name": "PartitionEpoch", "type": "int32", "versions": "0+",
          "about": "The expected epoch of the partition which is being updated. For legacy cluster this is the
ZkVersion in the LeaderAndIsr request." }
      ]}
    ]}
  ]
}
```

Response

The `TopicId` field is added to replace `TopicName` in the response. The response can return INELIGIBLE_REPLICA if any of the replicas in the new ISR is fenced or shutting down. The response can return NEW_LEADER_ELECTED if a new leader was elected after the partition state change. The response can return UNKNOW_TOPIC_ID when the topic id is unknown.

```
{
  "apiKey": 56,
  "type": "response",
  "name": "AlterPartitionResponse",
  // Version 1 adds LeaderRecoveryState field (KIP-704).
  //
  // Version 2 adds TopicId field to replace TopicName field, can return the following new errors:
  // INELIGIBLE_REPLICA, NEW_LEADER_ELECTED and UNKNOWN_TOPIC_ID (KIP-841).
   "validVersions": "0-2",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The top level response error code" },
    { "name": "Topics", "type": "[]TopicData", "versions": "0+", "fields": [
      { "name": "TopicName", "type": "string", "versions": "0-1", "ignorable": true, "entityType": "topicName",
        "about": "The name of the topic" },
      // New Field Begin
      { "name": "TopicId", "type": "uuid", "versions": "2+", "ignorable": true,
        "about": "The ID of the topic to alter ISRs for" },
      // New Field End
          { "name": "Partitions", "type": "[]PartitionData", "versions": "0+", "fields": [
        { "name": "PartitionIndex", "type": "int32", "versions": "0+",
          "about": "The partition index" },
        { "name": "ErrorCode", "type": "int16", "versions": "0+",
          "about": "The partition level error code" },
        { "name": "LeaderId", "type": "int32", "versions": "0+", "entityType": "brokerId",
          "about": "The broker ID of the leader." },
        { "name": "LeaderEpoch", "type": "int32", "versions": "0+",
          "about": "The leader epoch." },
        { "name": "Isr", "type": "[]int32", "versions": "0+", "entityType": "brokerId",
          "about": "The in-sync replica IDs." },
        { "name": "LeaderRecoveryState", "type": "int8", "versions": "1+", "default": "0", "ignorable": true,
          "about": "1 if the partition is recovering from an unclean leader election; 0 otherwise." },
        { "name": "PartitionEpoch", "type": "int32", "versions": "0+",
          "about": "The current epoch for the partition for KRaft controllers. The current ZK version for the
legacy controllers." }
      ]}
    ]}
  ]
}
```

Cluster Metadata Records

RegisterBrokerRecord

The `InControlledShutdown` field is added to the `RegisterBrokerRecord` to persist when a broker starts the controlled shutdown procedure.

```
{
  "apiKey": 0,
  "type": "metadata",
  "name": "RegisterBrokerRecord",
  "validVersions": "0-1",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "BrokerId", "type": "int32", "versions": "0+", "entityType": "brokerId",
      "about": "The broker id." },
    { "name": "IncarnationId", "type": "uuid", "versions": "0+",
      "about": "The incarnation ID of the broker process" },
    { "name": "BrokerEpoch", "type": "int64", "versions": "0+",
      "about": "The broker epoch assigned by the controller." },
    { "name": "EndPoints", "type": "[]BrokerEndpoint", "versions": "0+",
      "about": "The endpoints that can be used to communicate with this broker.", "fields": [
        { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
          "about": "The name of the endpoint." },
        { "name": "Host", "type": "string", "versions": "0+",
          "about": "The hostname." },
        { "name": "Port", "type": "uint16", "versions": "0+",
          "about": "The port." },
        { "name": "SecurityProtocol", "type": "int16", "versions": "0+",
          "about": "The security protocol." }
    ]},
    { "name": "Features", "type": "[]BrokerFeature",
      "about": "The features on this broker", "versions": "0+", "fields": [
      { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
        "about": "The feature name." },
      { "name": "MinSupportedVersion", "type": "int16", "versions": "0+",
        "about": "The minimum supported feature level." },
      { "name": "MaxSupportedVersion", "type": "int16", "versions": "0+",
        "about": "The maximum supported feature level." }
    ]},
    { "name": "Rack", "type": "string", "versions": "0+", "nullableVersions": "0+",
      "about": "The broker rack." },
    { "name": "Fenced", "type": "bool", "versions": "0+", "default": "true",
      "about": "True if the broker is fenced." },
    // New Field Begin
    { "name": "InControlledShutdown", "type": "bool", "versions": "1+", "default": "false",
      "about": "True if the broker is in controlled shutdown." }
    // New Field End
  ]
}
```

BrokerRegistrationChangeRecord

The `InControlledShutdown` field is added to the `BrokerRegistrationChangeRecord` as well.

```
{
  "apiKey": 17,
  "type": "metadata",
  "name": "BrokerRegistrationChangeRecord",
  "validVersions": "0-1",
  "flexibleVersions": "0+",
  "fields": [
   { "name": "BrokerId", "type": "int32", "versions": "0+", "entityType": "brokerId",
     "about": "The broker id." },
   { "name": "BrokerEpoch", "type": "int64", "versions": "0+",
     "about": "The broker epoch assigned by the controller." },
   { "name": "Fenced", "type": "int8", "versions": "0+", "taggedVersions": "0+", "tag": 0,
     "about": "-1 if the broker has been unfenced, 0 if no change, 1 if the broker has been fenced." },
   // New Field Begin
   { "name": "InControlledShutdown", "type": "int8", "versions": "1+", "taggedVersions": "1+", "tag": 1,
     "about": "0 if no change, 1 if the broker is in controlled shutdown." }
   // New Field End
  ]
}
```

Compatibility, Deprecation, and Migration Plan

The IBP and metadata.version will be bumped to gate changes to the RegisterBrokerRecord and the BrokerRegistrationChangeRecord records. The AlterPartition API version is picked based on the ApiVersions advertised by the controller.

INELIGIBLE_REPLICA is only returned for AlterPartition version >= 2; OPERATION_NOT_ATTEMPTED is used otherwise.

NEW_LEADER_ELECTED is only used for AlterPartition version >= 2; FENCED_LEADER_EPOCH is used otherwise.

Rejected Alternatives

None