KIP-589 Add API to update Replica state in Controller

Master KIP

KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum (Accepted)

Status

Current state: Accepted

Discussion thread: here

Voting thread: here

JIRA: KAFKA-9837

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

Motivation

Currently, log dir failure notifications are sent from the broker to the controller using a ZooKeeper watch. When a broker has a log dir failure, it will write a znode under the path `/log_dir_event_notification`. The controller watches this path for changes to its children. Once the watch is fired, the controller reads the data from all the children to get a list of broker IDs which had log dir errors. A LeaderAndIsr request is sent to all the brokers which were found in the notification znodes. Any broker which now has an offline replica (due to log dir failure) would respond with a storage error. This then causes the controller to mark the replica as offline and to trigger a leader election. This procedure is describe in detail in the original design KIP-112.

For the KIP-500 bridge release, brokers will be allowed to read from ZooKeeper, but only the controller will be allowed to write. Since we will not be able to use ZooKeeper as an event bus between brokers, we must come up with an alternate strategy.

With this KIP, we propose to add a new RPC that allows a broker to directly communicate state changes of a replica to the controller. This will replace the ZooKeeper based notification for log dir failures.

Public Interfaces

We will add a new RPC named AlterReplicaState which requires CLUSTER_ACTION permissions

```
AlterReplicaStateRequest => BrokerId BrokerEpoch EventType EventReason [Topic [PartitionId]]
        BrokerId => Int32
        BrokerEpoch => Int64
        NewState => Int8
        Reason => String
        Topic => String
        PartitionId => Int32

AlterReplicaStateResponse => ErrorCode [Topic [PartitionId ErrorCode]]
        ErrorCode => Int32
        Topic => String
        PartitionId => Int32
```

Possible top-level errors:

- CLUSTER_AUTHORIZATION_FAILED
- STALE_BROKER_EPOCH
- NOT_CONTROLLER
- UNKNOWN_EVENT_TYPE (new)

Partition-level errors:

- UNKNOWN_TOPIC_OR_PARTITION: The topic no longer exists.

Proposed Changes

Upon encountering errors relating to the log dirs, the broker will now send an RPC to the controller indicating that one or more replicas need to be marked offline. Since these types of errors are likely to occur for group of replicas at once, we will continue to use a background thread in ReplicaManager to allow these errors to accumulate before sending a message to the controller. The controller will synchronously perform basic validation of the request (permissions, check if topics exist, etc) and asynchronously perform the necessary actions to process the replica state changes.

Previously, the broker would only send its ID to the controller using the ZK watch mechanism. Since only the broker ID was sent (and not the broker epoch) the controller needed to probe the broker to learn its true state. Theoretically, a broker could have been repaired and restarted before the controller had a chance to react to the event. The controller would probe the brokers using LeaderAndIsr requests to learn in what state the replicas were. In this proposal, we now include the broker ID and epoch in the request, so the controller can safely update its internal replica state based on the request data. If the broker had, in fact, been restarted since sending the AlterReplicaState, the controller would be gated by the broker epoch and would not take any action.

RPC semantics

- The *EventType* field in the request is will only support the value 0x1 which will represent "offline"
- The *EventReason* field is a textual description of why the event is being sent

Failure Modes

Like the ZK watch before, the AlterReplicaState RPC from the broker to the controller is best-effort. We are not guaranteed to be able to send a message to the controller to indicate a replica state change. Also, since the processing of this RPC by the controller is asynchronous, we are not guaranteed that the subsequent handling of the state change will happen. The following failure scenarios are possible:

- If the broker cannot send the AlterReplicaState request to the controller and the offline replica is a follower, it will eventually be removed from the ISR by the leader.
- If the broker cannot send the AlterReplicaState request to the controller and the offline replica is a leader, it will remain the leader until the broker is restarted or the controller learns about the replica state through a LeaderAndIsr request (this is the current behavior).
- If the controller reads the AlterReplicaState and encounters a fatal error before handling the subsequent ControllerEvent, a new controller will eventually be elected and the state of all replicas will become known to the new controller.

For the Broker-side failures, we should implement retries on the AlterReplicaState messages.

Compatibility, Deprecation, and Migration Plan

Since this change is between the brokers, we will use the inter-broker protocol version as a flag for using this new RPC or not. Once the IBP has been increased, the brokers will not write out ZK messages for log dir failures and the controller will not set a watch on the parent znode. They will instead use the new RPC detailed here.

It's also worth mentioning that this KIP does not propose to be the exclusive mechanism for marking replicas offline. The controller will still mark replicas offline if it receives certain errors from a LeaderAndIsr response.

Rejected Alternatives

AlterIsr RPC

The AlterIsr RPC proposed in KIP-497 is similar in structure to the RPC proposed here. In the case of AlterIsr, it is used by the leader to indicate ISR changes to the controller. Since only the leader has up-to-date ISR information, a follower cannot reliably use this RPC to communicate an ISR change to the controller. It would require significant changes to the AlterIsr KIP to support this use case.

Specific RPC for log dir failures

Another rejected approach was to add an RPC which mirrored the JSON payload used by the ZK workflow currently implemented. This was rejected in favor of a more generic RPC that could be used for other purposes in the future. It was also rejected to prevent "leaking" the notion of a log dir to the public API and to the Controller.

Future Work

We have intentionally designed this KIP to accommodate future use cases involving the replica state. For example, we would potentially mark a replicas a "online" following some kind of log dir recovery procedure.

This RPC is quite similar to the existing ControlledShutdown RPC. If we extend this RPC to define a null list of topics to mean *all topics*, we could subsume the ControlledShutdown RPC with the AlterReplicaState RPC. After we implement this new RPC, we can consider if we want to move forward and consolidate ControlledShutdown into AlterReplicaState as a future KIP.