KIP-918: MM2 Topic And Group Listener

Status

Current state: *"Under Discussion"*

Discussion thread: *here*

JIRA: *KAFKA-14903*

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

Motivation

MM2 supports a dynamic topic and group filtering mechanism in which the replicated topics/groups can change when the list of available topics/groups change, or the filter configuration is updated. Currently, there is no way to access the list of topics and groups replicated by MM2.

The closest options right now are:

1. Using the ReplicationPolicy to find remote topics in the target cluster. This solution is not supported when IdentityReplicationPolicy is in use. Additionally, ReplicationPolicy work with the topic names, and do not take the latest topic filtering rules into account - e.g. if topic "test" was replicated at some point in AB, it created "A.test" in B. Then, if "test" was removed from the filter, the topic is not replicated anymore, but the replica topic "A.test" may still be present in B, listed as a remote topic.
2. Using the reported MirrorMakerMetrics. This solution only gives us an approximate, as certain topics might never appear due to not having any messages to be replicated, and also cannot properly tell when a topic was removed from the replication.

To address this, MM2 should support a TopicListener (nested in MirrorSourceConnector) and a GroupListener (nested in MirrorCheckpointConnector) plugin, and notify this plugin when the list of replicated topics/groups changes. This will allow users to follow the current set of replicated topics and groups, even with the IdentityReplicationPolicy.

Public Interfaces

New interfaces in :connect:mirror - TopicListener and GroupListener

```
interface TopicListener extends Configurable, AutoCloseable {
    void topicsChanged(Map<String, String> upstreamToDownstreamTopics);
}
interface GroupListener extends Configurable, AutoCloseable {
    void groupsChanged(List<String> replicatedGroups);
}
```

New default implementations:

1. DefaultTopicListener (NOP implementation of TopicListener)
2. DefaultGroupListener (NOP implementation of GroupListener)

New configurations:

1. MirrorSourceConnector - topic.listener.class (default: DefaultTopicListener) - Specifies which class to use as the TopicListener.
2. MirrorCheckpointConnector - group.listener.class (default: DefaultGroupListener) - Specifies which class to use as the GroupListener.

Proposed Changes

MirrorSourceConnector

1. At start, instantiate class defined by topic.listener.class config (topicListener).
2. In the refreshTopicPartitions method, when the list of replicated topics is computed, notify topicListener with the map of **upstream topic** - **replica topic names**.
3. In the stop method, close topicListener.

MirrorCheckpointConnector (almost the same changes as in MirrorSourceConnector, but with groups)

1. At start, instantiate class defined by group.listener.class (groupListener).
2. In the refreshConsumerGroups method, when the list of groups to checkpoint is computed, notify groupListener with the list of **group names**.
3. In the stop method, close groupListener.

Compatibility, Deprecation, and Migration Plan

No need for deprecation or migration, as it adds new capabilities, and the defaults of the new configurations do not change the current behavior. Existing users will not be impacted.

Test Plan

Unit tests should be sufficient to cover this feature.

Rejected Alternatives

Existing tools

As described in the motivation, the existing alternatives (using ReplicationPolicy or using MirrorMakerMetrics) does not fully fit the requirement.

Changing TopicFilter and GroupFilter

Theoretically, the existing filter interfaces could be changed - instead of acting as predicates on a single topic/group, they could accept a set of topic/group names, and filter the set. This would allow the filter plugins to act as the "listeners" of the current topic/group list.

The drawback of this would be that filters would not be aware of the additional filtering rules of the MM2 Connectors (e.g. internal mm2 topics are filtered by default), and could potentially report false positives. Additionally, mixing different filter implementations with different "listener" implementations would become tedious, since instead of using composition with 2 different plugins, users would need to implement custom plugins to get the desired combination of filter and listener implementations.