# KIP-710: Full support for distributed mode in dedicated MirrorMaker 2.0 clusters



## Status

**Current state**: *Accepted*

**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

KIP-382 introduced MirrorMaker 2.0, leveraging the Connect framework to provide a more reliable way of mirroring between Kafka clusters. The KIP also introduced the dedicated MirrorMaker 2.0 cluster, which is capable of running multiple Connect clusters and replication flows inside a dedicated cluster.

Unfortunately, the dedicated cluster mode has some missing features. There is an issue, which can render such clusters failing: the missing Connect REST API. MM2 clusters have this feature excluded, which makes follower->leader communication impossible in the cluster. Because of this, given certain workloads, the dedicated MM2 cluster can become unable to update task configurations. Another missing feature is the lazy resolution of configuration provider references in Connector configs - in dedicated mode, MM2 eagerly resolves such references, making it impossible to provide host-specific or sensitive configurations through the indirection of configuration providers.

The goal of this KIP is to solve these issues by enabling follower to leader communication in dedicated MirrorMaker 2.0 nodes, and lazily evaluating configuration provider references in Connector configs created by the MM2 dedicated mode.

### Failure scenario due to missing REST server feature

In MM2, dynamic configuration changes can occur based on TopicFilter and GroupFilter implementations. For example, the default TopicFilter implementation accepts topic name patterns to be included in the replication, enabling it to add new topics to an already running replication flow. (Similarly, dynamic config change can occur by providing an implementation of TopicFilter and/or GroupFilter which is capable of detecting changes during runtime.) The MirrorSourceConnector periodically polls these filters, picking up any changes in the mirrored topics and/or groups.

The issue occurs when a rebalance step results in the leader stopping its MirrorSourceConnector instance and moving it to a follower. Because of this, the leader is not capable of detecting topic or group filter configuration changes by itself. The follower running MirrorSourceConnector is capable of detecting the config change, but as per the Connect workflow, they cannot apply it to the current workload - they need to notify the leader through the REST API. Since REST is not available in dedicated MM2 mode, the notification never reaches the leader, rendering the cluster unable to detect config changes.

## Public Interfaces

- MM2 nodes now accept an *dedicated.mode.enable.internal.rest* config which is false by default.
- When *dedicated.mode.enable.internal.rest* is true, MM2 nodes now start a Connect REST server. This REST server has the internal Connect endpoints registered, where each endpoint is prefixed with the source and target of a replication flow. All the other Connect endpoints are excluded from the MM2 REST Server.
- When *dedicated.mode.enable.internal.rest* is true, MM2 worker id generation now uses the same pattern as the Connect worker generation (the advertised URL).
- MM2 now accepts all Connect REST properties to configure the REST server, with the exception of *rest.extension.classes*, *admin.listeners* and *admin.listeners.https.**
- MM2 now does not resolve configuration provider references in ConnectorConfigurations, letting those references to be resolved on each node of the cluster, at Connector configuration time.

## Proposed Changes

This KIP proposes 2 major changes:

- Enabling a single Connect REST server in the MirrorMaker 2.0 node, only supporting the internal Connect endpoints.
- Enabling the the lazy evaluation of configuration provider references in ConnectorConfigurations created by dedicated MirrorMaker 2.0 nodes.

### Enabling the Connect REST server

Currently, for each replication flow, a dedicated MM2 node creates a separate Connect node, and each node joins a Connect cluster dedicated to a specific replication flow. This is done by instantiating a worker and a herder. These two components are the essential parts of a classic Connect node. To support the full feature set of a Connect node, MM2 would also need to provide the Connect REST API between its nested Connect nodes.

This can be done by instantiating a REST server supporting the internal Connect endpoints. The API needs to be hierarchical to support endpoints separately for each replication flow. All endpoints will be prefixed with /source/target to separate the endpoints of the flows.

This would ensure that every nested Connect node has its full capabilities, and enables Connect follower nodes to communicate with their leaders.

In case REST is enabled, to also support the worker identification in the connect status topic, the worker id generation changes to: <*REST_HOST_NAME*>

To make sure that we keep the old behavior, the REST server has to be explicitly enabled by using a new config called *dedicated.mode.enable.internal.rest* , which is false by default.

### Allowing configuration provider references to be lazily resolved in ConnectorConfigurations created by MirrorMaker 2.0

Currently, when a dedicated MM2 node derives the base of the ConnectorConfiguration from its MirrorMakerConfiguration, it eagerly resolves all configuration provider references (e.g. environment variable references). Because of this, the configurations persisted in Kafka contain the resolved values instead of the references.

This is undesirable, as it makes it impossible to provide sensitive or host-specific configurations to the Connectors when using the dedicated mode of MM2. Connector configurations must be applicable on all hosts running members of the MM2 cluster. This can be problematic in case of file path type configurations, for example ssl keystore and truststore locations.

To solve this, MirrorMakerConfiguration should keep track of the original, pre-resolution state of the configuration, and use that to derive the ConnectorConfigurations. With this, the configs saved into Kafka will contain references instead of resolved values, and this also enables Connect's Worker to resolve the config provider references at a later point - matching the behavior of a vanilla Connect node.

## Compatibility, Deprecation, and Migration Plan

This KIP only extends the capabilities of MM2 nodes. Old behavior is kept by disabling the REST server by default. The config resolution change is not a breaking behavior change.

## Rejected Alternatives

### Extending the Connect framework

The discussion on KIP-382 brought up the an alternative solution of extending the Connect framework capabilities. Connect could leverage a feature of running connectors against multiple Kafka clusters in a single Connect cluster. This is the generalization of the dedicated MM2 feature: a Connect node can run multiple workers and herders, connecting to different Kafka clusters.

This solution would need significant changes in Connect and its REST API, while it is unclear if this feature would provide improvements in usage on-par with the amount of effort needed.

### Explicitly documenting that dedicated MM2 does not support distributed mode

Another possible solution is to explicitly reject the distributed mode of MM2, and document that it is not supported. Since the dedicated mode of MM2 provides essential improvements over the vanilla Connect mode (namely in terms of configuration and operations), distributed mode should be fixed instead of abandoned.