KIP-238: Expose Kafka cluster ID in Connect REST API

Status

Current state: Accepted

Discussion thread: here

JIRA: ⚠ Unable to render Jira issues macro, execution error. *(https://github.com/apache/kafka/pull/4314)*

Released: 1.1.0

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

Motivation

Connect's REST API exposes some information about the connect cluster itself, but does not expose any information about the Kafka cluster it is communicating with. If you have multiple Kafka clusters and possibly multiple Connect clusters, having this information is useful for a few different reasons. At its simplest it can be used to validate and/or discover where data is being read from/written to without requiring access to the Worker configuration. It also allows applications building on top of Kafka and Connect to associate the clusters, their metadata, connection info, etc, again without requiring access to the worker configurations.

Public Interfaces

Currently the `GET /` already describes some basic information about the connect worker:

```
{
  "version":"1.1.0-SNAPSHOT",
  "commit":"e5741b90cde98052"
}
```

This KIP will extend this to also include the Kafka cluster ID that the worker is configured to use (based on `bootstrap.servers` and any additional connection/security configurations in the worker configuration):

```
{
  "version":"1.1.0-SNAPSHOT",
  "commit":"e5741b90cde98052",
  "kafka_cluster_id":"I4ZmrWqfT2e-upky_4fdPA"
}
```

The naming is intentionally verbose, including a `kafka_` prefix to allow for a future `cluster_id` or `connect_cluster_id` to refer to a unique identifier for the Connect cluster.

Proposed Changes

The API is simple but the implementation does have some impact on the behavior of the worker on startup. Currently the values exposed in the API can be determined without connecting to external services. Additionally, the different Connect modes have the following startup behavior:

- Standalone - does not require a worker-level connection to the broker (only tasks require a producer or consumer), and therefore does not need to block on a connection to the cluster to be able to start serving HTTP requests
- Distributed - validates existence and configuration of config/offsets/status topics before starting the REST API HTTP server, therefore blocking the REST API until certain requests can be made. These requests timeout after the default request timeout unless the value is overridden (2 minutes)

This change will perform a synchronous lookup of the Kafka cluster information during startup of the `Worker` class, therefore blocking the REST API from serving requests until the broker can be contacted. Additionally, it will inherit the same default request timeout settings and therefore may timeout, throw an exception, and cause the worker to shutdown if the cluster cannot be contacted in time. This seems an acceptable tradeoff given distributed mode already has effectively the same limitation, it requires minimal access (metadata requests only), and it avoids having to handle intermediate states where we may not know the Kafka cluster ID (no null or sentinel values).

Simple unit testing of the `Worker` class and `RootResource` should be sufficient to validate the behavior.

Compatibility, Deprecation, and Migration Plan

This is a compatible change for users that can ignore extra fields in the payload. The Connect worker's principal must be able to make metadata requests on the cluster, which is already expected.

Rejected Alternatives

Technically, I believe it is possible to configure the worker, producer, and consumer to use different clusters (though I don't think this has ever been tested – only different security configurations/principals). There may not be a guarantee that the Kafka cluster ID is the same for the worker, source tasks, and sink tasks. We could expose all three of these, but in practice this seems like an odd edge case we should not be concerned with. If we ever encountered a need for it we could add further fields similar to the `kafka_cluster_id` field to handle the producer and consumer cases.