KIP-882: Kafka Connect REST API configuration validation timeout improvements

Status

Current state: *Under Discussion*

Discussion thread: *here*

Vote thread: *here*

JIRA: *here*

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

Motivation

Kafka Connect currently defines a default REST API request timeout of 90 seconds. If a REST API request takes longer than this timeout value, a `500 Internal Server Error` response is returned with the message "Request timed out".

The `POST /connectors` and the `PUT /connectors/{connector}/config` endpoints that are used to create or update connectors internally do a connector configuration validation (the details of which vary depending on the connector plugin) before proceeding to create or update the connector. If the configuration validation takes longer than 90 seconds, the connector is still eventually created after the config validation completes (even though a `500 Internal Server Error` response is returned to the user) which leads to a fairly confusing user experience.

Furthermore, this situation is exacerbated by the potential for config validations occurring twice for a single request. If Kafka Connect is running in distributed mode, requests to create or update a connector are forwarded to the Connect worker which is currently the leader of the group, if the initial request is made to a worker which is not the leader. In this case, the config validation occurs both on the initial worker, as well as the leader (assuming that the first config validation is successful) - this means that if a config validation takes longer than 45 seconds to complete each time, it will result in the original create / update connector request timing out.

Slow config validations can occur in certain exceptional scenarios - consider a database connector which has elaborate validation logic involving querying information schema to get a list of tables and views to validate the user's connector configuration. If the database has a very high number of tables and views and the database is under a heavy load in terms of query volume, such information schema queries can end up being considerably slow to complete.

Public Interfaces

The behavior of the existing `POST /connectors` and the `PUT /connectors/{connector}/config` endpoints will be modified in cases where the configuration validation exceeds the request timeout. Instead of proceeding with the connector create / update operation (which is the current behavior), we will abort the request.

Proposed Changes

After the configuration validation completes for a request to `POST /connectors` or `PUT /connectors/{connector}/config`, a check will be made to verify that the request timeout hasn't already been exceeded. If it has, the connector create / update request will be aborted.

Another change that will be made with this KIP is to avoid the double connector config validation issue in Connect's distributed mode. Workers will directly forward requests to create or update a connector to the leader without performing any config validation first. The only small benefit of the existing approach is that it avoids request forwarding to the leader for requests with invalid connector configs. However, it can be argued that it's cheaper and more optimal overall to forward the request to the leader at the outset, and allow the leader to do a single config validation before writing to the config topic. Since config validations are done on their own thread and are typically short lived operations, it should not be an issue even with large clusters to allow the leader to do all config validations arising from connector create / update requests (the only situation where we're adding to the leader's load is for requests with invalid configs, since the leader today already has to do a config validation for forwarded requests with valid configs). Note that the `PUT /connector-plugins/{pluginName}/config/validate` endpoint doesn't do any request forwarding and can be used if frequent validations are taking place (i.e. they can be made on any worker in the cluster to avoid overloading the leader).

Compatibility, Deprecation, and Migration Plan

The proposed changes shouldn't have any backward compatibility concerns outside of the unrealistic scenario where users are relying on the current behavior of connector create / update requests proceeding to completion even when config validation causes the request to exceed the timeout value. Note that this would still be possible by manually writing the connector's configuration to the Connect cluster's config topic.

Test Plan

- Add an integration test to verify that a connector is not created if config validation exceeds the request timeout.
- Add an integration test to verify that config validation only occurs a single time when requests to create or update a connector are made to a worker which is not the leader.
- Add unit tests wherever applicable.

Rejected Alternatives

Introduce a new internal endpoint to persist a connector configuration without doing a config validation

Summary: Instead of forwarding all create / update requests to the leader directly, we could do a config validation on the non-leader worker first and if the validations pass forward the request to a new internal-only endpoint on the leader which will just do the write to the config topic without doing a config validation first.

Rejected because: Introduces additional complexity with very little benefit as opposed to simply delegating all config validations from create / update requests to the leader. Furthermore, this could have security implications where the internal endpoint could be abused to bypass config validation (although the internal endpoint could potentially be secured using the mechanism introduced in KIP-507: Securing Internal Connect REST Endpoints).