KIP-980: Allow creating connectors in a stopped state

Status

Current state: *"Accepted"*

Discussion thread: *here*

Voting 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-875: First-class offsets support in Kafka Connect introduced a new `STOPPED` state for connectors along with some REST API endpoints to retrieve and modify offsets for connectors. Currently, only connectors that already exist can be stopped (or paused) and any newly created connector will always be in the `RUNNING` state initially. Allowing the creation of connectors in a `STOPPED` state will facilitate multiple new use cases. One interesting use case would be to migrate connectors from one Kafka Connect cluster to another. Individual connector migration would be useful in a number of scenarios such as breaking a large cluster into multiple smaller clusters (or vice versa), moving a connector from a cluster running in one data center to another etc. A connector migration could be achieved by using the following sequence of steps :-

1. Stop the running connector on the original Kafka Connect cluster
2. Retrieve the offsets for the connector via the `GET /connectors/{connector}/offsets` endpoint
3. Create the connector in a stopped state using the same configuration on the new Kafka Connect cluster
4. Alter the offsets for the connector on the new cluster via the `PATCH /connectors/{connector}/offsets` endpoint (using the offsets obtained from the original cluster)
5. Resume the connector on the new cluster and delete it on the original cluster

Note that this would require the connector to support offsets modification as described in KIP-875.

Another use case for creating connectors in a stopped (or paused) state could be deploying connectors as a part of a larger data pipeline before the source / sink data system has been created or is ready for data transfer.

Public Interfaces

"POST /connectors" REST API endpoint

A new optional field `"initial_state"` will be added to the request body format for the `POST /connectors` endpoint (the existing format can be seen in Kafka Connect's current OpenAPI specification). The new optional field `"initial_state"` can have a value of `"RUNNING", "PAUSED"` or `"STOPPED"` (case-insensitive). If the value of the `"initial_state"` field is invalid, a `400 Bad Request` response will be returned. If the field is omitted in the request body, the connector will be created in the `RUNNING` state by default (preserving the existing behavior). An example request body would look like:

CreateConnectorRequest

```
{
    "name": "file-sink-test",
    "config": {
        "connector.class": "org.apache.kafka.connect.file.FileStreamSinkConnector",
        "file": "test.txt",
        "topics": "test-topic",
        "tasks.max": "1"
    },
    "initial_state": "STOPPED"
}
```

Note that when a connector is created in a `PAUSED` or `STOPPED` state, no tasks will be spawned for the connector until it is resumed via the `PUT /connectors/{connector}/resume` endpoint. This method of creating connectors can currently be used in both the distributed mode as well as the standalone mode.

Standalone mode CLI

Kafka Connect in standalone mode is currently started using a command line like -

```
bin/connect-standalone.sh config/connect-standalone.properties [connector1.properties connector2.properties ...]
```

where the connector properties are Java Properties files containing String key-value connector configurations.

The CLI will be updated to also accept JSON files containing the connector configurations (i.e. the optional additional CLI arguments can be either Java Properties files or JSON files). The JSON files could be -

1. A simple JSON object containing only string key-value pairs representing the connector configuration (i.e. same as the existing request body format for the `PUT /connectors/{connector}/config` endpoint - see current OpenAPI specification).

   OR

2. A JSON object containing the `"name"` string, `"config"` object and (optionally) `"initial_state"` of the connector (i.e. same as the request body format for the `POST /connectors` endpoint)

Users won't need to explicitly specify the type of the connector configuration file, nor will we rely on the file name extensions. Instead, the file will be parsed "intelligently" - i.e. we'll first attempt to parse it into the above two JSON structures before attempting to parse it into Java Properties if the JSON parsing is unsuccessful. This order of parsing is chosen because the Java Properties format is very permissive and there won't be any errors on attempting to parse a JSON file into Java Properties. The "intelligent" parsing will ensure the smoothest possible user experience.

Allowing specifying connector configurations as JSON files to the standalone mode startup CLI offers two benefits. One is that users will be able to copy and use examples across both the methods of connector creation (REST API requests with JSON request bodies in standalone / distributed mode and JSON files passed to the standalone mode startup CLI). The second benefit is that any future extensions to connector creation requests would be easily applied across both the methods consistently.

Proposed Changes

Background

When a connector is created on a Kafka Connect cluster running in distributed mode via the `POST /connectors` endpoint, the connector's configuration is initially written to the cluster's config topic. A background thread on each Connect worker consumes from the config topic and when a new connector configuration is detected, a group rebalance is triggered which will result in the connector being assigned to a worker in the cluster and then started.

The `PUT /connectors/{connector}/pause` , `PUT /connectors/{connector}/resume` , `PUT /connectors/{connector}/stop` endpoints work by writing a target state to the config topic (see KIP-875 and KIP-52 for more background information) and can only be used for connectors that already exist in the cluster. Similar to the connector creation scenario, the target state is read asynchronously by the background thread on each worker and applied to the connector and its tasks. When a connector is created for the first time, there will be no target state initially and the connector is transitioned to the default target state of started / running.

In standalone mode, these operations occur essentially synchronously and the configs / target states are stored only in-memory.

Changes

When the leader Connect worker (distributed mode only) receives a create connector request (such requests made to non-leader workers are first forwarded to the leader internally) with a valid non-null `"initial_state"` field, it will write the appropriate target state to the config topic along with the connector configuration. If the Connect cluster is configured to use a transactional producer to write to the config topic (see KIP-618), the target state and the connector configuration will be written atomically in a single transaction. If the Connect cluster is not configured to use a transactional producer to write to the config topic, the target state will be written to the config topic before the connector configuration. This would ensure that in the unlikely but not impossible case that the leader worker dies between writing the two messages to the config topic, the connector won't be brought up in the wrong state. Target states for unknown connectors are essentially ignored so this order of writes is safer overall.

When a connector is created in a `PAUSED` or `STOPPED` state, the connector will be assigned to a worker post group rebalance and the connector thread will be created but the actual Connector instance won't be started. No tasks will be created for such connectors until they're resumed. The only difference between creating a connector in the `PAUSED` state versus the `STOPPED` state is that offsets modification will only be possible in the `STOPPED` state. There isn't any scenario where creating a connector in the `PAUSED` state is advantageous over creating it in the `STOPPED` state - however, since it's still a valid connector state and has been present for a lot longer than the `STOPPED` state, we allow it (`PAUSED` was introduced in AK 0.10.0.0 and `STOPPED` was introduced in AK 3.5.0).

The changes in standalone mode will be similar - the connector's target state will be updated prior to the connector being started if a valid `"initial_state"` is specified in the connector creation request (or at startup in a JSON file specified via command line arguments).

Compatibility, Deprecation, and Migration Plan

The new field being added to the request body format for the `POST /connectors` endpoint is optional and if omitted, the default behavior mirrors the existing behavior (i.e. the connector will be created in a `RUNNING` state). Thus, there aren't any backward compatibility issues introduced in this KIP. The target state written to the config topic will use the convention established in KIP-875 (`"state"` and `"state.v2"`) in order to facilitate cluster downgrades and rolling upgrades (older Connect workers may not recognize the `STOPPED` state).

Test Plan

Unit tests, integration tests or system tests (whichever is deemed most appropriate) will be added for these cases -

- Can create a connector in the `PAUSED` state in standalone mode
- Can create a connector in the `STOPPED` state in standalone mode
- Can create a connector in the `PAUSED` state in distributed mode
- Can create a connector in the `STOPPED` state in distributed mode
- No tasks are spawned for a connector created in the `PAUSED` or `STOPPED` state
- Creating a connector with no explicit state specified results in a `RUNNING` state connector
- Can resume a connector created in the `PAUSED` or `STOPPED` state
- Can delete a connector created in the `PAUSED` or `STOPPED` state
- Can modify the offsets for a connector created in the `STOPPED` state
- Can use JSON files containing connector key-value configurations with standalone mode
- Can use JSON files containing a JSON object similar to the request body format for the `POST /connectors` endpoint with standalone mode

Future Work

Allow specifying offsets directly during connector creation

In order to more directly support connector migrations / creating a connector with a specific starting offset, we could introduce an optional `"offsets"` field to the connector creation request body. Internally, this would write the specified offset before starting the connector. This would be a more streamlined way of doing things from the user's point of view as opposed to creating a connector in the `STOPPED` state, altering its offsets, and then resuming the connector.

Rejected Alternatives

Directly expose the existing target states as possible initial connector states

Summary: The current target states that can be written to the config topic are `STARTED` (when a connector is resumed), `PAUSED` and `STOPPED` and we could use these as valid values for the new `"initial_state"` field in the connector creation request body.

Rejected because: These target states aren't currently directly exposed to the users and the states that are actually exposed to the users via the `GET /connectors/{connector}/status` endpoints are: `UNASSIGNED` , `RUNNING` , `PAUSED` , `FAILED` , `RESTARTING` and `STOPPED`. Using `RUNNING` over `STARTED` as a valid value for the new `"initial_state"` field in the connector creation request body makes more sense since that's the state that users are already familiar with (the `PAUSED` and `STOPPED` states are common across and aren't an issue).

Make similar changes to the `PUT /connectors/{connector}/config` REST API endpoint

Summary: The `PUT /connectors/{connector}/config` endpoint allows either creating or updating a connector (depending on whether it already exists or not). We could allow specifying the connector state in this endpoint as well.

Rejected because: It isn't particularly useful to allow modifying the state of a connector along with updating its configuration since both those operations can already be done separately (whereas with the connector creation case, it isn't currently possible to modify the state of a non-existent connector) and the `POST /connectors` endpoint already covers the connector creation case. Furthermore, the `PUT /connectors/{connector}/config` endpoint currently accepts a map of connector key-value configurations as its request body (OpenAPI specification for reference), and adding a new `"initial_state"` (or equivalent) field would require significantly modifying the request body format and supporting two different request body formats for the endpoint. We could potentially use a request parameter or request header to specify the state, but it would be fairly unusual to have two completely different ways of specifying the connector state across two endpoints (and it seems more appropriate to include the state in the request body anyway since it's a part of the request itself rather than metadata or a modifier).

Only allow creating connectors in RUNNING / STOPPED states

Summary: Since there is no real advantage to creating connectors in a `PAUSED` state versus the `STOPPED` state, disallow creating connectors in the `PAUSEI` state to reduce the surface area of the API.

Rejected because: The `PAUSED` state is still a valid connector state and has been present for a lot longer than the `STOPPED` state (`PAUSED` was introduced in AK 0.10.0.0 and `STOPPED` was introduced in AK 3.5.0). It would likely be more confusing for users if we treated `PAUSED` as an invalid `"initial_state"` in the connector creation request body.