KIP-891: Running multiple versions of a connector

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here*

Motivation

Kafka connect currently supports running a single version of a connector. As the externals system that a connector supports evolves over time, the connectors may need to make incompatible changes between versions to keep up with new capabilities. Without the ability to run multiple versions of a connector, we would need to run the connectors in two different connect clusters if we want to support both version simultaneously. In addition, support for multiple connectors makes it possible to do two phase upgrades to connectors with the benefit of easier rollbacks. A connector runtime restart will still be required to install a newer version, however existing connectors can still continue to use the older version. The migration to the newer version is decoupled from the installation of the connector and will not require a cluster restart. The decoupling also makes rollbacks easier and will no longer require a cluster rollback. Finally, multiple version eases development toil while making changes to a connector.

Public Interfaces

Connector version will be provided using a new connector configuration parameter, named `connector.version` . The naming scheme can be debated but it should be indicative of the context in which it applies and not have potential for existing connector conflicts. The connect runtime will interpret this the new config in the following ways.

- If `connector.version` is present then the runtime will try to initialise with the given version and fail otherwise.
- If `connector.version` is absent or set to `latest` the runtime will use the latest version of the connector available. This preserves existing behaviour.

The connector-plugins endpoint that retrieves the config of a connector, `/{plugin-name}/config` will be updated to include a `version` query parameter which will default to latest. This API returns the set of configurations for a given connector. Since between versions the configurations can change its allow a user given version to return the correct configs. The version is added as a query parameter, for example -`/S3SinkConnector/config?version=v1.1.1`

Proposed Changes

Running multiple versions requires two primary capabilities to be present in the runtime.

- Isolation of connector version. A version of a connector and its dependencies should not interfere with another version.
- A versioning mechanism for connectors.

Fortunately both of these are already present in the connect runtime.

Plugin Loading Isolation

Plugin loading isolation introduced in KIP-146 - Classloading Isolation in Connect uses a different classloader for each plugin listed under the plugin path. The plugin loading mechanism identifies each subdirectory under the `plugins.path` directory listing as a separate plugin. Any connector jars and associated dependencies under the subdirectory are isolated from other connector and dependencies. This mechanism is generic and considers the same connector under two different plugin directory as two different plugins. This extends to all classes and dependencies provided along with the connector jar (including custom transformation that some connectors have). This feature allows two versions of the same connector to be loaded in isolation given that they are under two different plugin locations, which is by default the recommended installation process. Indeed, even now, multiple versions of a connector are recognised and identified by the runtime, however only the latest version is ever loaded during connector initialisation. This will be updated to allow the other recognised connector plugins to be loaded based on the version.

Connector Versioning

The connect API defines the Connector abstract class as implementing a Versioned interface. Each connector implementation can override the inherited versioned method to provide a version in code to the runtime. The exact version of the connector could be statically defined by directly overriding the method or could be injected in the connector jar with java build tools during artefact creation. The runtime already uses the value provided here to order connector plugins and determine the latest version of a connector. No updates will be required to this module. The comparison of version is based on maven artefact versioning. It is a generic versioning scheme that supports semantic, alphabetical and combinations with support additional modifiers (like alpha, beta, release and snapshot build versions). The javadoc here has more details on the implementation.

Summarising the versioning feature below.

- mixing of '`-`' (hyphen) and '`.`' (dot) separators,
- transition between characters and digits also constitutes a separator: `1.0alpha1 => [1, 0, alpha, 1]`
- unlimited number of version components,
- version components in the text can be digits or strings,
- strings are checked for well-known qualifiers and the qualifier ordering is used for version ordering. Well-known qualifiers (case insensitive) are:
    - `alpha` or `a`
    - `beta` or `b`
    - `milestone` or `m`
    - `rc` or `cr`
    - `snapshot`
    - `(the empty string)` or `ga` or `final`
    - `sp`

  Unknown qualifiers are considered after known qualifiers, with lexical order (always case insensitive),
- a hyphen usually precedes a qualifier, and is always less important than something preceded with a dot.

The work here is augmentative, primarily to the plugin loading mechanism. The set of changes is more or less laid out in this draft PR I did as a POC.

The existing `connector-plugins` endpoint lists all the versions of a connector. A user can easily identify which version of a connector is currently running through this, when not directly present in the configs, as then it is always the latest version in the listing.

Compatibility, Deprecation, and Migration Plan

Any existing connector without version information in the configuration will continue to default to using the latest version available in the plugins path. Any cluster updates with a later version of the connector installed will result in the connector also getting updated to the latest version. This preserves the existing behaviour.

A connector can be updated to assign it a particular version, and beyond this point the connector will always run with the assigned version (provided the version is present) even across cluster upgrades. However cluster downgrades to older connector runtimes will end up using the older version of the connector. Since the assignment of version to a connector needs to be explicitly opted in by the user through a connector update, this limitation is easily evident.

Test Plan

Integration and Unit tests will be added.

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.