KIP-769: Connect APIs to list all connector plugins and retrieve their configuration definitions

Status

Current state: Accepted

Discussion thread: *here*

JIRA: *KAFKA-13510*

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

Motivation

When starting a connector, users must provide the connector configuration. The configuration often also includes configurations for other plugins such as SMTs or converters. Today, Connect does not provide a way to see what plugins are installed apart from connectors. This make it difficult for users building data pipeline to know which plugins are available and what is possible. Basically they have to know how the Connect runtime is set up. Even once they know the plugins that are available, they then have to go look at the plugins documentation or, in the worst case, look directly at the source code to find their configuration definitions.

Connector plugins should be discoverable via the REST API. Their configuration definitions should also be easily retrieved. This would significantly ease the process of building pipelines and enable building tools and UIs that can manage Connect data pipelines.

Public Interfaces

- `GET /connector-plugins`: This endpoint will be updated to allow listing all plugins. The response structure of the objects in the array remain unchanged. A new query parameter "`connectorsOnly`" will be added and it will default to true so it's fully compatible with the current behavior. Users will be able to list all Connectors, Transformations, Converters, HeaderConverters and Predicates plugins by setting it to false. Classes that implement multiple plugin types will appear once for each type. For example SimpleHeaderConverter will be listed as a converter and as a header_converter. Possible values for the "type" field are "sink", "source", "converter", "header_converter", "transformation" and "predicate".

  For example GET `/connector-plugins?connectorsOnly=false` will return:

```
[
  {
    "class": "org.apache.kafka.connect.file.FileStreamSinkConnector",
    "type": "sink",
    "version": "3.2.0"
  },
  {
    "class": "org.apache.kafka.connect.file.FileStreamSourceConnector",
    "type": "source",
    "version": "3.2.0"
  },   {
    "class": "org.apache.kafka.connect.converters.ByteArrayConverter",
    "type": "converter"
  },
  {
    "class": "org.apache.kafka.connect.transforms.Cast$Value",
    "type": "transformation"
  },
  {
    "class": "org.apache.kafka.connect.transforms.predicates.HasHeaderKey",
    "type": "predicate"
  },
  {
    "class": "org.apache.kafka.connect.storage.SimpleHeaderConverter",
    "type": "header_converter"
  },
  {
    "class": "org.apache.kafka.connect.storage.SimpleHeaderConverter",
    "type": "converter"
  },
  ...
]
```

Currently only Connector plugins are versioned, so we won't include the version field for other plugins.

- `GET /connector-plugins/<plugin>/config`: This new endpoint will return the configuration definitions of the specified plugin. It will work with all plugins returned by `/connector_plugins`.

  The plugin can be specified via its fully qualified class name or its Connect alias like in the existing `/connector-plugins/<plugin>/config/validate` endpoint. If a plugin does not override the `config()` method, the response is an empty array.

  For example, accessing http://localhost:8083/connector-plugins/org.apache.kafka.connect.transforms.Cast$Value/config will return:

```
[
  {
    "name": "spec",
    "type": "LIST",
    "required": true,
    "default_value": null,
    "importance": "HIGH",
    "documentation": "List of fields and the type to cast them to of the form field1:type,field2:type to cast
fields of Maps or Structs. A single type to cast the entire value. Valid types are int8, int16, int32, int64,
float32, float64, boolean, and string. Note that binary fields can only be cast to string.",
    "group": null,
    "width": "NONE",
    "display_name": "spec",
    "dependents": [],
    "order": -1
  }
]
```

Proposed Changes

REST API:

- A new path will be added to `ConnectorPluginsResource` to retrieve the plugin configuration definitions

```
@GET
@Path("/{plugin}/config")
public List<ConfigKeyInfo> getPluginConfig() {}
```

- Listing connector plugin will accept an optional query parameter "`connectorsOnly`" that defaults to `true`

```
@GET
@Path("/")
public List<ConnectorPluginInfo> listConnectorPlugins(@DefaultValue("true") @QueryParam("connectorsOnly")
boolean connectorsOnly) {}
```

Converter interface:

Add a `config()` method to `Converter` with a default implementation.

```
public interface Converter {

[...]

    /**
     * Configuration specification for this set of converters.
     * @return the configuration specification; may not be null
     */
    default ConfigDef config() {
        return new ConfigDef();
    }
}
```

It's common for custom converters to implement both `Converter` and `HeaderConverter`. As the 2 methods to retrieve the `ConfigDef` will have exactly the same signature, it will still be possible to implement both interfaces.

Compatibility, Deprecation, and Migration Plan

- `/connector-plugins` keeps its current behavior and will only expose the new behavior when a new query parameter is set.
- When accessing `/connector-plugins/<plugin>/config` on existing converters that don't implement the `config()` method, an empty array will be returned. If a converter is also implementing `HeaderConverter`, and hence already have a `config()` method, it will be automatically used and the config will be returned.
- `/connector-plugins/<plugin>/config` is a new endpoint that doesn't cause compatibility issues.

I propose to flip the query parameter value to list all plugins by default in the next major release.

Rejected Alternatives

- Add a new endpoint /plugins for listing all plugins: It would be confusing to list both worker and connector plugins together. We'd then end up with 3 endpoints, /plugins, /worker-plugins and /connector-plugins which is as confusing!
- Group connectors by type when listing them: This would break compatibility with the existing /connector-plugins behavior. As it's a very commonly used endpoint, it's preferred to keep compatibility.
- Add a new endpoint /worker-plugins to list worker plugins (Rest Extensions and Config Providers): The use case is to allow administrators to check the plugins installed in each worker. Connect shouldn't expose worker internal details to all users and it's not clear what information would be useful for admins. Also Connect already has a /admin endpoint which should be reused for admin tasks.
- Make all plugins implement Versioned. Initially we wanted to make all plugins consistent, but this either force having a default implementation for version() which would allow Connectors to not implement it, or force introducing another interface (PossiblyVersioned) to version other plugins which did not make a lot of sense since version does not have any contract today.