KIP-898: Modernize Connect plugin discovery

Status

Current state: *Released in 3.6.0*

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

When starting a Connect worker, the worker must discover all of the plugin classes available on the class path and plugin.path. This is necessary for advertising the set of supported plugins via the Connect REST API, and allowing short plugin aliases to be used in place of full class names.

Currently, that discovery consists of “plugin.path scanning” in which the worker reflectively loads every class on the class path and plugin.path. This has a number of downsides:

- Delays the startup of the worker
  - Slows down provisioning time of a worker in cloud environments
  - Increases downtime after an unexpected worker failure
  - Increases the duration of cluster rolls
  - Inflates runtime of tests which make use of connect clusters
- Increases the memory footprint of a worker
  - Defeats lazy-loading optimizations for static resources
  - Increases the fixed cost of a connect worker
  - Reduces the memory otherwise available for plugin workloads
- Increases the bug and security surface-area of a worker
  - Relies on an out-of-date reflections library
  - Runs code that would otherwise be inactive or unreachable

It is for these reasons that it is desirable to replace the current implicit declaration paradigm with an explicit declaration paradigm which is simpler to evaluate at worker start-up.

Proposed Changes

Instead of scanning for plugin subclasses among every potential class, the worker will read from ServiceLoader manifests and module info during startup.

This mechanism is already used for the `ConfigProvider` , `ConnectRestExtension` , and `ConnectorClientConfigOverridePolicy` plugins, and the loading of these plugins will continue as before. This change will apply the mechanism to the `SinkConnector` , `SourceConnector` , `Converter` , `HeaderConverter` , `Transformation` , and `Predicate` plugins for the first time.

Public Interfaces

Plugins

This change will require connector developers to add service declarations to their plugins. This can be done in two ways:

- Add one or more ServiceLoader manifest files.
- Add a `module-info.java` containing one or more `provides … with` directives.

    - See Section 7.7.4 of the Java Language Specification 9 for details
    - Connectors using module-info files *without* also providing a ServiceLoader manifest will not be accessible in Java 8 environments.

Once a service declaration is added, plugins can be released and distributed normally.

This includes plugins built and published by the Kafka project itself, which will have ServiceLoader manifest files added as part of implementing this KIP.

Connect Worker Configuration

The Connect Worker will have a new configuration: `plugin.discovery` which can take one of multiple values with the following meanings:

- `ONLY_SCAN` : Corresponding to the legacy behavior, in which every class on the plugin.path is scanned on startup for plugins. In addition, a warning will be printed to suggest reconfiguring the worker to HYBRID_WARN. This is intended to disable the new code paths if a bug is present.
- `HYBRID_WARN` : In addition to the legacy scanning behavior, use the new mechanism and print a warning if a class is present via scanning but not via ServiceLoader. If there are no discrepancies, a warning will be printed to suggest reconfiguring the worker with `SERVICE_LOAD` . This is intended to inform operators that they are depending on out-of-date plugins that need to be updated.
- `HYBRID_FAIL` : Same as `HYBRID_WARN` , except a discrepancy between the old and new mechanisms will cause a worker to fail to start up, instead of appearing at runtime as a missing plugin. This is intended for use in downstream unit and packaging tests to assert that all plugins have been updated.
- `SERVICE_LOAD` : Only use the new ServiceLoader mechanism to load plugins. This is intended for production usage after all plugins have been updated, and will be the only mode with performance benefits.

The default value for this configuration will be `HYBRID_WARN` .

The default value for this configuration when used in the `EmbeddedConnectCluster` test utility will be `HYBRID_FAIL` .

Plugin Path Management Script

In addition, a new script `bin/connect-plugin-path.sh` will be developed to manage the worker plugin path. For the purposes of this migration, this script will execute plugin path scanning and generate shim JARs which include `ServiceLoader` manifests. This can be run ahead-of-time during CI, and will allow a connect instance to use non-updated plugins with `SERVICE_LOAD` .

The script would take the following arguments, with the following meanings:

- A positional argument `sub-command` which takes exactly one of the following values:
    - `sync-manifests`
        - For each concrete plugin implementation which is missing a ServiceLoader manifest, add a manifest file and/or entry.
            - For a single `jar` or `zip` file, for which a new resource file may be added to the existing archive.
            - For a directory with an arbitrary hierarchy of `jar` or `zip` files, for which additional directories and/or files may be added to the existing directory.
            - For a directory with an arbitrary hierarchy of `class` files, for which additional directories and/or files may be added to the existing directory.
        - The path and all subdirectories and files specified in other options must be writable.
    - `list`
        - Print a human-readable summary of a plugin path and the plugins contained within.
        - For each plugin, include:
            - The fully qualified class name
            - Plugin aliases (if available)
            - The version (if available)
            - Whether the class is discoverable via scanning
            - Whether the class is discoverable via ServiceLoader
        - The path and all subdirectories and files specified in other options must be readable.
- `--plugin-location <single-plugin-jar-zip-dir>`
    - The value of this argument will be a single plugin, which can be any of the following:
        - a single `jar` or `zip` file
        - a directory with an arbitrary hierarchy of `jar` or `zip` files
        - a directory with an arbitrary hierarchy of `class` files
    - This can be specified zero or more times.
- `--plugin-path <list-of-paths>`
    - The value of this argument will follow the same semantics of the worker properties `plugin.path` configuration.
    - This will be equivalent to specifying multiple `--plugin-location` arguments, one for each top-level archive, and for each immediate sub-folder of each top-level directory.
    - This can be specified zero or more times.
- `--worker-config <worker-properties-file>`
    - From this worker properties file, the plugin.path contents on-disk will be mutated.
    - This will be equivalent to extracting the `plugin.path` configuration from the worker properties and specifying `--plugin-path` .
    - This can be specified zero or more times.
- `--dry-run`
    - Can only be specified in combination with `sync-manifests.`
    - If specified, execute all of the steps needed for the normal script execution except the final writing the changes to disk.
    - If not specified, the prescribed changes to the plugin path are written to disk.
    - If a condition which would prevent the script from completing normally is detected, the exit code of the script will be non-zero, and details will be printed via stderr in a human-readable format.
- `--keep-not-found`
    - Can only be specified in combination with `sync-manifests.`
    - If specified, ServiceLoader manifests for classes which cannot be found will not be removed

- If not specified, ServiceLoader manifests for classes which cannot be found will be removed
- If not specified, any manifest files or JARs which become empty due to class name removal will be removed

The `list` command is meant for inspecting the plugin.path before and after a migration takes place, and should expose the information that the `sync-manifests` command is using to perform the migration.

This script would migrate the specified paths in-place, and require the input files to be writable. The arguments which do not require a worker config are intended to provide smaller subunits of the migration for callers which want to divide the migration for error handling, modular CI builds, or reusable tooling.

If the script fails at any point with `sync-manifests` but without `--dry-run` specified, the plugin path may be left in an indeterminate state and should not be relied upon for correctness. After a script failure, it is recommended to clear the disk contents and restore it to a known-good state. If used in CI, a script failure can be made to cause the build to fail and be retried from the beginning.

If the script succeeds, subsequent runs on the same script directory should be idempotent. Subsequent runs on a partially changed directory should be idempotent for the unchanged parts, and should re-migrate the changed parts.

If a plugin declares an implementation via a module-info.java file, a duplicate shim manifest will not be generated. If a plugin declares an implementation via a module-info.java which is not loadable, the module-info.java will not be modified.

This script will only list and migrate plugins which are on the plugin.path of a Connect worker, and are loaded in isolation. This script will not list or migrate plugins which are included on the classpath, and will assume that classpath plugins have manifests added through some other method.

Compatibility, Deprecation, and Migration Plan

Developers

At any time, including before this KIP's vote passes or the feature is merged to the upstream, connector plugins can be updated to include ServiceLoader manifests. These manifests will be inactive, and not affect the functionality of the Connect worker.

Once a plugin developer updates their test runtime to a version with this feature, they will have test failures to notify them if they are noncompliant. As a temporary workaround, they can change the mode to `HYBRID_WARN` to allow their build to complete. After they add the necessary plugin manifests, their tests will pass. They can leave the test configuration at `HYBRID_FAIL` , or use `SERVICE_LOAD` for more performant tests execution.

Operators

Once a connect operator updates their environment to a version with this feature, they will receive log warnings. If they notice these warnings, they will be able to upgrade plugins to versions which alleviate the warning, or contact their vendors/plugin developers to encourage them to update their plugins. They will be able to see the progress of this update in the startup logs, and via `bin/connect-plugin-path.sh list`.

While waiting for plugins to update, they can use the `bin/connect-plugin-path.sh sync-manifests` to migrate plugins at the point of use in their CI, and use `SERVICE_LOAD` mode in their environment configuration.

After a connect operator has updated all of the plugins, they can remove `bin/connect-plugin-path.sh sync-manifests` from their CI, and change the CI test configuration to `HYBRID_FAIL` to catch any regressions.

Deprecation

The new interfaces will ideally be released in a 3.x version of Kafka. None of the existing or new interfaces will be deprecated immediately.

In a follow-up KIP as early as 4.0, we should propose changing the configuration default to `SERVICE_LOAD`, given the ease of applying the workarounds. That KIP should also decide on a deprecation schedule for the plugin path scanning behavior, and deprecate the necessary configuration values.

In a second follow-up KIP as early as 5.0, we should schedule the removal of the scanning behavior. This would mean that connector plugins built for Kafka <3.x will not work for Kafka 5.0, or whatever version the removal takes place. Plugins with manifests will work for versions of Kafka <4.0 without issue.

Test Plan

Existing system tests will be configured to start workers with `SERVICE_LOAD` immediately for performance reasons, and as this is now the recommended running mode.

New system tests will be written to exercise each of the configuration values to confirm that a fully-migrated cluster will start up successfully in all modes.

New non-migrated, systems-test-only plugins will be added to the system test build to verify that a non-migrated plugin will have the intended effect in each mode. These plugins will be used to test the `bin/connect-plugin-path.sh` script. As part of this, existing system-test-only plugins will be refactored out of the publicly distributed build, and special cases for them removed from production code-paths.

Manual testing with several existing publicly available connectors will confirm that non-Apache plugins behave in a similar way to Apache test plugins used in automated tests.

Rejected Alternatives

- Using OSGi. In addition to the reasons noted in KIP-146, OSGi represents a much more invasive change to the Connect framework than this KIP is targeting, and with much less clear benefit. There are also existing plugins using the ServiceLoader paradigm which would require extra migrations.
- Have the migration script copy-on-write and not mutate the on-disk worker config or plugin.path. This adds a lot of complexity to the script, and complexity in specifying the output locations. This is easily avoided by the user copying their plugins to a writable scratch space before running the migration script.
- Discarding ClassLoaders to enable garbage collection of scanned classes after scanning is complete. This solves the ongoing memory overhead of the scanned classes, but does not remove the initial CPU overhead of the scanning operation itself.
- Adding module-info.java files for Apache plugins, in addition to, or in lieu of, ServiceLoader manifests. Because Kafka supports Java 8, we cannot rely on Java 9+ features. And adding modules for Kafka is outside the scope of this improvement, and deserves attention in a separate KIP.