KIP-75 - Add per-connector Converters

Status

Current state: Adopted

Discussion thread: *here*

JIRA: *KAFKA-3845*

Released: 0.10.1.0

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

Motivation

Today, Kafka Connect uses a single set of `Converters` for the entire cluster (set in the `WorkerConfig`). This choice was motivated by the idea that usually a single cluster will be working with one format because most users will want to standardize on a single data format, e.g. keep all their data in Avro, JSON, etc., and if they use a single format, we can reduce configuration overhead by using an automatic default. However, inevitably users will end up with a need for a different format in a few cases, whether it be due to legacy data & systems, legacy connector systems, or simply ease of integration. Occasionally users may want to use Connect with a different data format, but today that requires running a separate process or cluster.

This KIP proposes to allow overriding `Converters` on a per-connector basis. Utilizing the default should still strongly be encouraged for the benefits listed above, but this provides an escape hatch to avoid having to run a whole new cluster for one connector with different requirements.

Public Interfaces

The key addition are two configuration options to `ConnectorConfig` (note that this is inherited by `SourceConnectorConfig` and `SinkConnectorConfig`, and while not public interface, the Connector configuration options are):

- `key.converter` - class to instantiate as the `Converter` for keys created by/passed to this connector
- `value.converter` - class to instantiate as the `Converter` for values created by/passed to this connector

Additionally, any configurations namespaced under the prefixes `key.converter.` and `value.converter.` will be passed to the `configure` methods of the instantiated `Converters`. This allows important converter-specific configurations to be passed to the `Converter` and mirrors the existing behavior of `Converter` configs in `WorkerConfig`.

Proposed Changes

The new configuration options will have `null` values by default, in which case the `Worker`-level `key.converter` and `value.converter` settings will take effect. There will be no other changes to the behavior of the connectors and framework.

Compatibility, Deprecation, and Migration Plan

This is an addition that has no compatibility, deprecation, or migration concerns. The override behavior on a per-converter basis does not imply that the worker/cluster-level default will ever be removed or deprecated.

Test Plan

We already have unit tests and system tests that validate different converter settings for `Workers`. Adding one unit test (validating correctness within the `Herder`/`Worker`) and one system test (validating the use end-to-end) should sufficiently verify the new behavior. Existing tests that will not be modified will verify that the existing `Worker`-level default continues to behave as expected.

Rejected Alternatives

None