KIP-901: Add connectorDeleted flag when stopping tasks

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: ⚠ Unable to render Jira issues macro, execution error.

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

Motivation

This KIP is an extension of KIP-883: Add isDeleted flag when stopping a connector, and focuses on providing a flag indicating that a connector has been deleted when the task is stopped. This flag can then be used by the Task implementation to perform any additional cleanup steps if needed. This KIP is related to (and supersedes) KIP-419: Safely notify Kafka Connect SourceTask is stopped.

As connectors interact with external systems, they sometimes need to provision external resources. Imagine a Sink connector that creates new queues in a Messaging System before writing messages to them, or a Source connector that activates an account before polling a source system, or any other use cases that requires extra setup. A more concrete example (and one that concerns us in particular) is a source connector that audits database changes by creating an "audit" table and setting up database triggers to capture database row-level updates into the audit table.

In cases like these we might want to cleanup the resources when the connector that provisioned them is deleted. There can be many reasons why cleanup is desirable: be it to save costs (e.g. the external system charges per active account) compute resources (triggers writing to an audit table that will no longer be polled should be removed, and so does the audit table), and many others.

Public Interfaces

org.apache.kafka.connect.connector.Connector

```
package org.apache.kafka.connect.connector;

public interface Task {
    /**
     * Get the version of this task. Usually this should be the same as the corresponding {@link Connector}
class's version.
     *
     * @return the version, formatted as a String
     */
    String version();

    /**
     * Start the Task
     * @param props initial configuration
     */
    void start(Map<String, String> props);

    /**
     * Stop this task.
     */
    void stop();

    /**
     * Stop this task, while also indicating that the task is being stopped because the connector was
     * deleted.
     * <p>
     * Tasks are not required to override this method, unless they need to perform additional cleanup
     * actions in cases where the connector has been deleted.
     *
     * @param connectorDeleted indicates if the connector has been deleted.
     */
    default void stop(boolean connectorDeleted) {
        stop();
    }
}
```

Proposed Changes

Add an overload to the `void stop(boolean connectorDeleted)` method to the Task public API, with a default implementation that calls the existing `void stop()` method. This new method can then be overridden by connectors that need to take any additional steps as part of the deletion process.

Herder classes

Both StandaloneHerder and DistributedHerder invoke methods from the `Worker` class to start/stop the tasks.

For the `StandaloneHerder`, the tasks will be stopped with the **`connectorDeleted`** flag set to **true** as part of the `StandaloneHerder#deleteConnectorConfig(...)` method. In the case of the `DistributedHerder`, the **`connectorDeleted`** flag will be computed during the `RebalanceListener#onRevoked(...)` callback, by checking if the tasks being revoked are for a connector that has been deleted (the connector configuration no longer exists in the ClusterConfigState store).

Compatibility, Deprecation, and Migration Plan

The proposed change is fully backward-compatible with existing Kafka Connect releases. The new method added to the public interface includes an default implementation of the new method, so existing connectors don't need to override it if not needed.

Test Plan

Integration tests will be added to make sure that the new flag is used when stopping a task for a deleted connector.

Rejected Alternatives

Delete provisioned resources out-of-band

In theory, users can monitor Kakfa Connect configuration topics to determine if/when a connector has been deleted. Reacting to this event outside of the connector's context is probably not very useful, as there might not be enough contextual information to perform any meaningful action. There is some value on keeping these concerns encapsulated within the connector framework itself.