KIP-419: Safely notify Kafka Connect SourceTask is stopped

Status

Current state: *Under Discussion*

Discussion thread: https://lists.apache.org/thread.html/597d794a2a43b2568224d7db3a5c832e2166d2825ed7bf95102ceb25@%3Cdev.kafka.apache.org%3E

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

When a Kafka Connect SourceTask is being stopped, the stop() method is called to signal that the task is to stop. However, this method is not necessarily the final method called and the stop() method can actually be called more than once in some situations for the same SourceTask. If the SourceTask has resources that it needs to clean up, there is no safe, easy-to-use way to achieve this. This KIP proposes the addition of a new stopped() method to the SourceTask interface which is guaranteed to be the final call in the SourceTask's lifecycle meaning that is can safely be used for cleaning up resources.

There is no need for an equivalent change on SinkTask because the shape of that interface makes it more clear when the task is safe to clean up its resources safely.

Public Interfaces

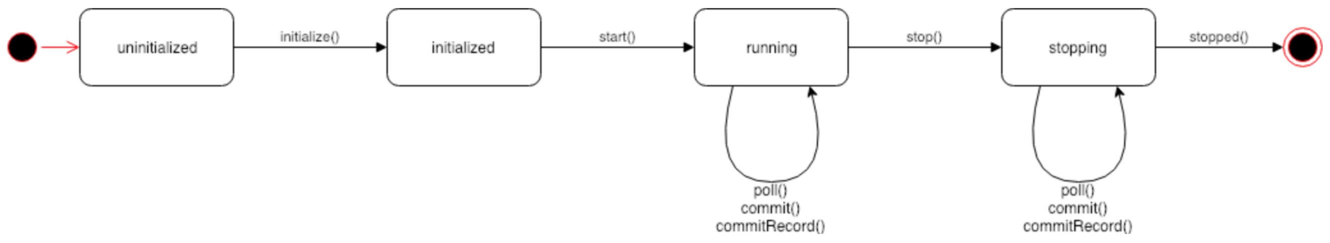
A new callback method is added to the SourceTask interface.

```
public abstract class SourceTask implements Task {

    /**

     * <p>

     * This task has stopped and can safely release resources.

     * </p>

     * <p>

     * SourceTasks are not required to implement this functionality. This hook is provided

     * for systems that have resources such as network connections that need to be

     * released safely once the SourceTask has indeed stopped.

     * </p>

     */

    public void stopped()

    {

        // This space intentionally left blank.

    }

}
```

Proposed Changes

Add the new stopped() callback to the SourceTask interface. In the Kafka Connect runtime, call this method as the final call to the SourceTask interface once it is known that all activity on the task has indeed stopped.

Here's a diagram of the states of the SourceTask.



The states are not formally part of the code, but there are essentially 4 states:

- uninitialized - the SourceTask has been instantiated but not initialized. When initialize() is called by Kafka Connect, the SourceTask moves into initialized state.
- initialized - the SourceTask has been initialized but not started. When start() is called by Kafka Connect, the SourceTask moves into running state.
- running - the SourceTask has been started. Kafka Connect regularly calls poll() to receive records from the source system. As each record is acknowledged by Kafka (or discarded), Kafka Connect calls commitRecord(). Periodically on another thread, Kafka Connect calls commit(). When stop() is called by Kafka Connect, the SourceTask moves into stopping state.
- stopping - The stop() call is intended to interrupt a blocking call to poll(), but it's not necessarily the case that poll() is blocking when it is called. So, any of poll(), commit() and commitRecord() can still be called. Usually, an active call to poll() completes and the current batch completes processing, followed by a final call to commit(). However, commit() is also running on another thread so the precise sequence of calls is a bit variable. When it's all quiesced, stopped() is called by Kafka Connect and the SourceTask can release all resources.

The addition in this KIP is just the call to stopped().

Compatibility, Deprecation, and Migration Plan

- This is a new optional callback that connectors do not have to implement. No migration concerns.

Rejected Alternatives

- Modify the Kafka Connect runtime so that stop() is called just once.