KIP-558: Track the set of actively used topics by connectors in Kafka Connect

Status

Current state: *Adopted.*

Discussion thread: here

Vote thread: here

JIRA: here

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

Motivation

As more and more applications, deployments and integrations are being built around Kafka Connect, it'd be useful for users, operators and applications to know the set of topics that a connector has used since it was first created. Today, for sink connectors this information may be extracted more easily, by looking at the configuration of the connector. But given that a sink connector configuration possibly describes subscription to topics through regular expressions, figuring out exactly which topics a connector is using requires listing and matching such topics explicitly, for example with command line tools that will match topics and will also confirm that such topics have records. For source connectors, it is not practical at this time to use Connect alone to figure out with accuracy the set of topics the connector has been using at any point in time. Bookkeeping of active topics requires some sort of external tracking by the connector or its user. Keeping track of the topics that a source connector uses gets harder when the connector is allowed to create such topics (either by enabling automatic topic creation, or by using a version that supports KIP-158).

Public Interfaces and Proposed Changes

In order to allow Kafka Connect to keep track of the set of topics that a specific connector has used during its lifetime the following public-facing changes are proposed. It's worth noting that the required changes do not include any changes to the public interfaces or classes of the Kafka Connect framework and rebuilding existing connector code is not required in order to use this feature.

Storing and returning names of topics that are actively used by connectors

Information regarding the topics a connector is using can be stored as a sequence of records in Kafka. This approach would be also consistent with Connect's design principle to use Kafka as its main dependency in order to track progress and store status updates.

Connect's internal status topic (set by the configuration property `status.storage.topic`) already fulfils the specification requirements of a topic that could be used to store information related to topic observability. Specifically:

- It's a compacted topic.
- It's a partitioned topic (i.e. has more than one partitions). Order is retained within a single partition, but global order is not required.
- It's a topic that is being fully read by every worker during startup.
- Every worker listens to updates and is able to read new messages written to this topic.
- Records are keyed.
- Older workers can skip messages that don't understand in this topic.
- Json encoding is used for keys and values.

Given that, it's suggested here to reuse the `status.storage.topic` in order to store the topics that each connector is actively using in addition to the connector and tasks information that is currently being stored in this topic.

Format of the new status record

When a worker task first detects that a connector is producing to or consuming from a Kafka topic, the worker task will append a record to the status topic with the following format:

Key format	Key example	Value format	Value example

`status-topic-${topic-name}: connector-${connector-name}`	`status-topic-foo:connector-some-source`	`{`
		`  "topic": {`
		`  "name": string,`
		`  "connector": string,`
		`  "task": int32,`
		`  "discoverTimestamp": int64`
		`  }`
		`}`
			`{`
			`  "topic": {`
			`  "name": "foo",`
			`  "connector": "some-source",`
			`  "task": 0,`
			`  "discoverTimestamp": 1579297899`
			`  }`
			`}`

The topic name can be safely separated by the connector name because the delimiter `:` (colon character) is not a valid topic name character in Kafka.

Connect workers reading this topic are computing a map from connector names to sets of topic names. This map represents an up-to-date view of which connectors are using which topics at any time based on the key of these records and whether the value entry of the record is null or non-null. If the most recent record value for a specific topic and a specific connector is non-null, then this topic is considered part of the set of topics that this connector is using. If the most recent record has a null value (is therefore a tombstone record), then the connector is not using this topic anymore. Tombstone records are interpreted by Connect workers as deletion actions of previous topic status records for this connector.

The information stored in the value of the Kafka record is selected to include the topic name, the connector name, the task ID of the task that last reported the topic is used by the connector and a timestamp relative to when this topic was detected as active. While the record value is not essential to decide whether a topic is used by a connector and it is partially redundant compared to what is stored in the key, it makes these entries easier to read and follow as well as more useful during runtime or when troubleshooting the topics used by a connector. Whereas the Kafka record key includes the topic name and the connector name, the Kafka record value stores additionally the ID of the task that succeeded to store a topic status record last (in case more than one task produces a record concurrently for a short period of time) and a timestamp to mark when this topic was detected as active. In the future the record value can be easily extended to include additional information.

Compared to the existing keys for the status topic, which currently have prefixes `status-connector-` and `status-task-`, the new key format extends the set of status topic record keys in a readable and intuitive way by adding the prefix `status-topic-` to the keys of the new Kafka records.

Recording active topics

When a connector is configured to run in a Connect cluster for the first time, its set of active topics is empty. Nevertheless, when its tasks start processing their first records (records of type `SourceRecord` for source tasks and records of type `SinkRecord` for sink tasks), the worker will start inspecting the Kafka topic of each of these records. If the worker detects that a topic does not belong to the set of active topics for this specific connector, it produces a record to the status topic with the format described in the previous section. The worker in the Connect cluster maintains an up-to-date view of all the connectors' active topics by continuously reading the `status.storage.topic`.

It is expected that, multiple Connect workers may compete to append more than one record to the `status.storage.topic`. These records will have the same key and because the topic is compacted, all the Kafka records of a specific key will eventually collapse into a single entry. As soon as a worker detects the addition of a topic to a connector's set of active topics, the worker will not post to the `status.storage.topic` additional update records for the connector and this newly-detected active topic.

Resetting a connector's set of active topics

Tracking the set of active topics for a connector as described above implies that such sets are monotonically increasing in size over time. However, during the lifetime of a connector, some topics might stop to be actively used (e.g source tasks no longer produce to that topic, or sink tasks don't have any new records to consume from a topic). For this reason, this KIP proposes to introduce an explicit Connect REST API request to reset a connector's set of active topics.

When a Connect worker receives this request, it sends a tombstone message for each topic in a connector's set of active topics. It's worth noting that this operation is not atomic with respect to the whole set of topics. Kafka provides atomicity at the record level and it guarantees that two records (e.g. a tombstone record and a topic status record) will be atomically appended to the log. Therefore, the order in which this may happen if a request to reset a connector's set of active topics is interleaved with actual production or consumption of records from the connector's tasks is not characterized by a *happens-before* relationship between reset (production of tombstone message) and recording (production of a non-tombstone message) actions.

Resetting the set of active topics of a connector while this connector is running is fine, as long as the intention is to reset any topics that are no longer used by the connector and retain the ones that are active. Topic reset is a composable operation with respect to a connector's normal execution. Soon after the reset, the worker tasks will populate the `status.storage.topic` with new topic status messages for the topics that the connector is currently using.

Restarting, reconfiguring or deleting a connector

Just restarting a connector (without altering the configuration) has no effect on the recorded set of active topics for that connector.

Reconfiguring a connector also has no effect on the recorded set of active topics. For sink connectors, that means that a topic that was included in the previous configuration of the connector but is not included in the current configuration, will still show up in the set of active topics. An explicit reset request will have to be issued (immediately after the connector is reconfigured) in order to remove these old topics from the connectors active set. This requirement is suggested in this KIP to keep the symmetry between source and sink connectors with respect to reset and also keep the new as well as the existing Connect REST API endpoints simple and with a focused mission.

Deleting a connector will reset this connector's set of active topics even when resetting topic tracking for connectors has been disabled. Successful reset of the history of topics used by connectors does depend on whether the connector was gracefully deleted. Partial reset can be followed by another attempt to reset tracking of topics for a connector.

Extensions to Connect REST API

The existing Connect REST API will be extended to include two new endpoints.

- A `GET /connectors/{name}/topics` endpoint. This endpoint will return the set of topic names that the connector has been using since its creation or since the last time its set of active topics was reset.
  There's no defined order in which the topics are returned and consecutive calls to this endpoint may return the same topic names but in different order.
  Here's an example of a GET request and its returned payload.

Get the set of active topics from a connector called 'some-source'

```
$ curl -s 'http://localhost:8083/connectors/some-source/topics' | jq
{
  "some-source": {
    "topics": [
      "foo",
          "bar",
          "baz",
    ]
  }
}
$
```

- A `PUT /connectors/{name}/topics/reset` endpoint. This endpoint will reset the set of topic names that the connector has been using since its creation or since the last time its set of active topics was reset.
  When a PUT request is issued and resetting connector active topics is allowed, the request returns an HTTP Status 200 with no payload. Success of resetting the history of a topic usage is independent of whether a connector is still running (this request can be reapplied after the deletion of the connector).

Successful reset of the set of active topics of a connector called 'some-source'

```
$ curl -X PUT -s 'http://localhost:8083/connectors/some-source/topics/reset' | jq
$
```

When a PUT request is issued but resetting the set of active topics for connectors is disabled, the request return HTTP Status of 403 with the following error message:

Attempt to reset the set of active topics when reset is not allowed

```
$ curl -X PUT -s 'http://localhost:8083/connectors/some-source/topics/reset' | jq
{
  "error_code": 403,
  "message": "Topic tracking reset is disabled"
}
$
```

Configuration

Topic tracking will be enabled by default, as will the ability to reset the set of active topics for a connector. However, the following two configurations give the administrator of the Connect cluster the ability a) to disable this feature altogether or b) to enable topic tracking but without allowing calls to the Connect REST API to reset the history of active topics for a connector.

	Type	Default	Possible Values	Description
`topic.tracking.enable`	boolean	true	true, false	Whether the Connect worker will track and persist which topics are actively used per connector. It's highly recommended to set the same value in all the workers of a Connect cluster

`topic.tracking.allow.reset`	boolean	true	true, false	Whether to allow requests to reset the set of active topics for specific connectors.

Security

This feature enables a user or application to find out the topic names that are used by a connector. With respect to security, this feature inherits the security characteristics that similar functionality has in Kafka Connect at the moment. Specifically:

- A user that has the ability to query the status, create or reconfigure a connector via the Connect REST API will be able to get the set of topic names that a connector uses. If access to specific endpoints is restricted for certain users, the Connect cluster administrators should consider restricting access to the new endpoints in a similar way.
- The topic names that a connector uses, are persisted in the `status.storage.topic`. This is an existing internal topic for Kafka Connect. Administrators should restrict access to the sets active topics per connector in the same way that they currently restrict access to the configuration and the status of connectors in their Connect clusters.

Given the above, the implementation of this KIP does not require extra steps to secure access to the set of active topic names that connectors are using.

Compatibility, Deprecation, and Migration Plan

This KIP introduces a new feature. It's only effect on compatibility is related to the persistence of a new type of records in Connect's `status.storage.topic`. If a Connect cluster is downgraded to a version that does not understand the records introduced in this KIP a warning will be printed in the logs per record.

In order to upgrade a Connect cluster to a version that supports this KIP, upgrading and restarting the Connect workers is the only requirement. Upgrading the workers can happen in any order and rolling restarts are allowed but not required. It is advised to set the worker configuration relevant to this feature the same on every worker of the Connect cluster.

Rejected Alternatives

Few options that could be used but ultimately are not included in this proposal are the following:

1. Expose topic tracking information for each connector programmatically by extending the `Connector` abstract class. Implementation of topic tracking and observability through the Connect REST API seems adequate to support a rich set of use cases.
2. Expose topic tracking information for each connector as a metric. This solution is not as flexible and easy to consume as introducing a Connect REST API endpoint. Additionally, Connect has already been preferring exposing similar information via its REST API instead of jmx. It seems preferable for this KIP to continue on that path.
3. Introduce a new, separate, internal topic to record topic tracking information. This seems unnecessary at this point because the `status.storage.topic` is a low volume topic that already shares the same specifications with the topic that would be required by this feature. Keeping topic tracking information in the `status.storage.topic` keeps the migration and compatibility implications very lightweight.