KIP-158: Kafka Connect should allow source connectors to set topic-specific settings for new topics

Status

Current state: Under Discussion

Discussion 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 of 0.11.0.0, Kafka Connect can automatically create its internal topics using the new AdminClient (see KIP-154), but it still relies upon the broker to auto-create new topics to which source connector records are written. This is error prone, as it's easy for the topics to be created with an inappropriate cleanup policy, replication factor, and/or number of partitions. Some users would rather not configure their brokers with auto.create.topics.enable=true, and in these cases users must manually pre-create the necessary topics. That, of course, can be quite challenging for some source connectors that choose topics dynamically based upon the source and that result in large numbers of topics.

Kafka Connect should instead be able to create the topics automatically for source connectors, using a replication factor, number of partitions, and other topic-specific settings declared in a source connector configuration. If these properties are not specified, the previous Connect behavior of relying upon the topics to exist or be auto created by the broker. Additionally, operators of Connect clusters should be able to either enable or disable this feature.

This feature does not affect sink connectors and does not change the topic-specific settings on any existing topics.

Public Interfaces and Proposed Changes

This proposal defines a simple way for source connector configurations to specify whether topics to which the source connector will write should be created by Connect if those topics do not already exist. Additionally, this feature is disabled by default for the whole Connect cluster, though it can be enabled via a new Connect worker configuration property.

Worker Configuration

In order to use this feature, the Connect cluster operator must configure the configurations for all Connect workers in the cluster with `topic.creation.enable=true`. Even then, the feature will only be used for source connectors whose configuration specifies the default topic attributes for new topics (see below).

This proposal **adds** one new Connect worker configuration, which must be set identically on all workers in the Connect cluster:

Property	Type	Default	Possible Values	Description
`disable.topic.creation.for.connectors`	boolean	true	true, false	Whether the Connect worker should allow source connector configurations to define topic creation settings. When 'true', source connectors can use this feature. When 'false', new source connector configurations that use these `topic.creation.*` configs would error, while these configs would be ignored (and a warning reported) for previously-registered source connector configs that used these properties.

Source Connector Configuration

This proposal **adds** several source connector configuration properties that specify the default replication factor, number of partitions, and other topic-specific settings to be used by Connect to create any topic to which the source connector writes that does not exist at the time the source connector generates its records. None of these properties has defaults, so therefore this feature is enabled for this connector only when the feature is enabled for the Connect cluster and when the source connector configuration specifies at least the replication factor and number of partitions.

Property	Type	Default	Possible Values	Description
`topic.creation.default.replication.factor`	int	n/a	>= 1 when a value is specified	The replication factor for new topics created for this connector. This value must not be smaller than the number of brokers in the Kafka cluster.
`topic.creation.default.partitions`	int	n/a	>= 1 when a value is specified	The number of partitions new topics created for this connector.
`topic.creation.default.${kafkaTopicSpecificConfigName}`		n/a		Any of the Kafka topic-level configurations for the version of the Kafka broker where the records will be written. The broker's topic-level configuration value will be used if that configuration is not specified for the rule.

Note that the Kafka topic-level configurations does vary by Kafka version, so source connectors should specify only those topic settings that the Kafka broker knows about.

Also, these properties have no effect if the feature is disabled on the Connect cluster via `topic.creation.enable=false` in the cluster's worker configurations.

Examples

The replication factor and number of partitions must be specified in the source connector configuration to enable topic creation for the connector. The following shows an example of these properties specifying for all new topics created by Connect for this connector a replication factor of 3 and 5 partitions:

Portion of an example source connector configuration using topic creation rules

```
...
topic.creation.default.replication.factor=3
topic.creation.default.partitions=5
...
```

The source connector configurations can optionally specify Kafka topic-level configurations to override the broker's defaults for new topics. The following shows an example specifies the `compact` cleanup policy, a minimum of 2 in-sync replicas, and no unclean leader election:

Portion of an example source connector configuration using topic creation rules

```
...
topic.creation.default.replication.factor=3
topic.creation.default.partitions=5

topic.creation.default.cleanup.policy=compact
topic.creation.default.min.insync.replicas=2
topic.creation.default.unclean.leader.election.enable=false
...
```

Sink Connector Configuration

This feature **does not** affect sink connectors or their configuration.

REST API

The existing Connect REST API includes several resources whose request and response payloads will be affected by this proposal, although the structure of those payloads are already dependent upon the specific type of connector. Applications that use the REST API must already expect such variation, and therefore

Security

When topic creation is enabled in the Connect worker, the worker may attempt to create topics to which the source connector(s) write that are not known to exist. The Admin API allows the Connect worker to request these topics be created, but will only attempt to create topics that do not already exist.

Therefore, in order to use this feature, the Kafka principal specified in the worker configuration and used for the source connectors (e.g., `producer.*`) must have the permission to DESCRIBE and CREATE topics. If the worker's producer does not have the necessary privileges to DESCRIBE existing and CREATE missing topics but a source connector does specify the `topic.creation.*` configuration properties, the worker will log a WARNING and will default to the previous behavior of assuming the topics already exist or that the broker will auto-create them when needed.

Note that when the Connect worker starts up, it already has the ability to create in the Kafka cluster the internal topics used for storing connector configurations, connector and task statuses, and source connector offsets. If creating topics is not desired for security purposes, this feature should be disabled.

Compatibility, Deprecation, and Migration Plan

When users upgrade an existing Kafka Connect installation, they **do not** need to change any configurations or upgrade any connectors: this feature will be enabled but as previously-registered source connector configurations would not include any `topic.creation.*` configuration properties, ***Kafka Connect will behave exactly as before*** by assuming the topics exist or else will be auto-created by the broker.

After upgrading, users **must** alter the configuration of any source connector to enable the creation of new topics, by adding the `topic.creation.default.replication.factor` and `topic.creation.default.partitions` properties plus optionally other `topic.creation.default.*` properties.

This feature **will not** affect source or sink connector implementations, as the connector API is unchanged and running connectors have no exposure to this feature. It also **does not** change the topic-specific settings on any existing topics.

Finally, this feature uses Kafka's Admin API methods to check for the existence of a topic and to create new topics. This feature will do nothing if the broker does not support the Admin API methods, which is equivalent to relying upon auto-topic creation. If ACLs are used, the Kafka principal used in the Connect worker's `producer.*` settings is assumed to have privilege to create topics when needed; if not, an error will be logged but the worker will revert to the old behavior of assuming the topics exist or will be auto-created by the broker.

Rejected Alternatives

Several alternative designs were considered but ultimately rejected:

1. Change only the Java API and have no configuration changes. This very simple approach would have required no changes to a connector configuration yet still given the source connector tremendous flexibility and responsibility in defining the topic-specific settings for each new topics (e.g., using the Admin API). This approach was rejected because it still relies upon the connector implementation to address/handle all variation in topic-specific settings that might be desired between new topics; because connector users have very little control over the topic-specific settings; and because the connector to be modified to take advantage of the new feature and would therefore not work with older connectors.
2. Change the Java API and use connector configuration properties to define the topic-specific settings used as defaults on all topics. This approach is a bit more flexible than the first alternative in that it allows for connector users to specify some default topic-specific settings in configuration properties. However, this approach was rejected because it offers connector users very little flexibility since it still relies upon the source connector to determine the settings for each of the topics.
3. Change the Java API and use connector configuration properties to define the topic-specific settings using rules that apply different settings to different topics. This approach was proposed in an earlier version of this KIP, but discussion highlighted that this was optimizing for the exceptional case where source connectors wrote to many topics and those topics needed different replication factors, number of partitions, and/or topic-specific settings. This resulted in a very complex configuration that was thought to be useful in a very small number of cases. It also exposed connectors to a new Java API, but again this would require changes in the source connector implementations and would restrict the Connect versions on which those connectors could be deployed.
4. Allow the connector to modify the topic-specific settings on an **existing** topic. This can be complicated, since not all topic settings can be easily changed. It also would introduce potential conflicts between a connector and other admin clients that are attempting to change the topic configuration settings to different values. Such a scenario would be extremely confusing to users, since they might not expect that the source connector is configured to modify the topic settings for an existing topic.
5. Should `topic.creation.default.replication.factor` have a default value? A default replication factor of 3 is a sensible default for production, but it would fail on small development clusters. By making this property be explicit, users that are configuring source connectors have to choose a value that makes sense for their Kafka cluster. It also has the advantage that not having a default means that this property is required to enable topic creation on a source connector, and this obviates the need for a separate `topic.creation.enabled` in the connector configuration.
6. Should the default value for `topic.creation.default.replication.factor` take into account the current number of brokers? Doing so would be very brittle and subject to transient network partitions and/or failed brokers, since the *actual* number of brokers might be smaller than the replication factor assumed by the user creating the connector configuration, and the user would have no feedback that a topic was created with fewer replicas than desired.
7. Should the `topic.creation.default.partitions` have a default value? The only sensible default is 1, and that's not always very sensible.
8. Should the Connect worker have a new `disable.topic.creation.for.connectors` property? This property allows operators of a Connect cluster to prevent source connectors from even using this feature. It would be possible (albeit more complicated) to not have the worker configuration property and to instead expect operators to use ACLs and instead give the Connect worker's producer CREATE topic permissions.