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

Status

Current state: Adopted

Current active discussion thread: here

Previous discussion threads: here and here

Vote thread (current): here

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

Released: *AK 2.6.0*

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 Kafka clusters disable auto topic creation via `auto.create.topics.enable=false`, and in these cases users creating connectors 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, as well as other topic-specific settings declared in a source connector configuration. Additionally, after the introduction of KIP-458, Kafka Connect may create source connector topics by optionally using connector-specific Kafka client settings that are declared in the source connector's configuration using appropriate overrides. 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.

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 enabled by default for new Connect workers, though it can be disabled via a new Connect worker configuration property. The proposed topic creation during runtime is relevant only to source connectors and it does not affect sink connectors. It also does not change the topic-specific settings on any existing topics.

Worker Configuration

The proposed feature is enabled by default and its activation is controlled by the single configuration property `topic.creation.enable`. 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`.

After an upgrade and by explicitly switching this configuration to `true`, this feature will only be used for source connectors whose configuration specifies at least the replication factor and number of partitions for at least one group, as described 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
`topic.creation.enable`	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.*` properties 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 for at least one group. Users may choose to use the default values specified in the Kafka broker by setting the replication factor or the number of partitions to -1 respectively.

Different classes of configuration properties can be defined through the definition of groups. Group definition is following a pattern that resembles what has been used previously to introduce property definition for [transformations in Kafka Connect](). The config property groups are listed within the property `topic.creation.groups`. The hierarchy of groups is built on top of a single implicit group that is called **default**. The **default** group always exists and does not need to be listed explicitly in `topic.creation.groups` (if it does, it will be ignored with a warning message).

Property	Type	Default	Possible Values	Description
`topic.creation.groups`	List of String types	empty	The group **default** is always defined for topic configurations. The values of this property refer to additional groups	A list of group aliases that will be used to define per group topic configurations for matching topics. If the feature if topic configs is enabled, The group **default** always exists and matches all topics.
`topic.creation.$alias.include`	List of String types	empty	Comma separated list of exact topic names or regular expressions.	A list of strings that represent regular expressions that may match topic names. This list is used to **include** topics that match their values and apply this group's specific configuration to the topics that match this inclusion list. $alias applies to any group defined in `topic.creation.groups` but not the **default**
`topic.creation.$alias.exclude`	List of String types	empty	Comma separated list of exact topic names or regular expressions	A list of strings that represent regular expressions that may match topic names. This list is used to **exclude** topics that match their values and refrain from applying this group's specific configuration to the topics that match this exclusion list. $alias applies to any group defined in `topic.creation.groups` but not the **default**. Note that exclusion rules have precedent and override any inclusion rules for topics.
`topic.creation.$alias.replication.factor`	int	n/a	>= 1 for a specific valid value, or -1 to use the broker's default value	The replication factor for new topics created for this connector. This value must not be larger than the number of brokers in the Kafka cluster, or otherwise an error will be thrown when the connector will attempt to create a topic. For the **default** group this configuration is required. For any other group defined in `topic.creation.groups` this config is optional and if it's missing it gets the value the **default** group
`topic.creation.$alias.partitions`	int	n/a	>= 1 for a specific valid value, or -1 to use the broker's default value	The number of partitions new topics created for this connector. For the **default** group this configuration is required. For any other group defined in `topic.creation.groups` this config is optional and if it's missing it gets the value the **default** group
`topic.creation.$alias.${kafkaTopicSpecificConfigName}`	several	broker value		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. $alias applies to the **default** as well as any group defined in `topic.creation.groups`

Note that these configuration properties will be forwarded to the connector via its initialization methods (e.g. `start` or `reconfigure`). Also note that the [Kafka topic-level configurations]() do vary by Kafka version, so source connectors should specify only those topic settings that the Kafka broker knows about. Topic settings rejected by the Kafka broker will result in the connector failing with an exception, to avoid silently ignoring invalid topic creation properties.

If the connector fails to create a topic for any reason, the task that attempts to create that topic will be stopped and will have to be manually restarted once the issue that resulted in it failure is resolved. Given that topic creation is supported during runtime, such failures are expected to happen any time during the lifetime of a connector and not only during its initial deployment.

The configuration properties that accept regular expressions accept regex that are defined as [Java regex]().

Topic configs have always at least one group, the **default** group. This group has as required config properties the replication factor and the number of partitions. Also it has an implicit inclusion list that matches all topics and an implicit exclusion list that is empty. Therefore, configuring these two properties for the **default** topic config group is not required and will be ignored (with a warning message in the logs).

In terms of configuration composition between groups and precedence order:

- A topic might get its configuration properties from one or more groups
- Groups are listed in order of preference within the property `topic.creation.groups`, with the highest preference group first and the lowest priority group last. The **default** does not have to be explicitly listed and has always the lowest priority. Given that preference is uniquely defined by the sequence of groups in `topic.creation.groups`, the order in which group configurations occur within a java properties file or a json encoded configuration does not matter.

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.

Configuration Examples

The following are examples that demonstrate the additions in the configuration of source connectors that this KIP is proposing. For simplicity, these examples show only snippets of the connector configuration properties that deal with topic creation and they are shown in Java properties format. The replication factor and number of partitions must be specified at least for the **default** group in the source connector configuration in order to enable topic creation for the connector.

Example 1: All new topics created by Connect for this connector will have replication factor of 3 and 5 partitions. Since **default** is the only group of topic creation properties, the config `topic.creation.groups` can be skipped:

Portion of an example source connector configuration using topic creation rules

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

Example 2: By default, new topics created by Connect for this connector will have replication factor of 3 and 5 partitions with the exception of topics that match the inclusion list of the *inorder* group, which will have 1 partition:

Portion of an example source connector configuration using topic creation rules

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

topic.creation.inorder.include=status, orders.*
topic.creation.inorder.partitions=1
...
```

Example 3: By default, new topics created by Connect for this connector will have replication factor of 3 and 5 partitions, while the `key_value_topic` and `another.compacted.topic` topics or topics that begin with the prefix `configurations` will be compacted and have a replication factor of 5 and 1 partition.

Portion of an example source connector configuration using topic creation rules

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

topic.creation.compacted.include=key_value_topic, another\\.compacted\\.topic, configurations.*
topic.creation.compacted.replication.factor=5
topic.creation.compacted.partitions=1
topic.creation.compacted.cleanup.policy=compact
...
```

Example 4: By default, new topics created by Connect for this connector will have replication factor of 3 and 5 partitions, while topics that begin with the prefix `configurations` will be compacted. Additionally, topics that match match the inclusion list of `highly_parallel` and don't match its exclusion list will have replication factor of 1 and 1 partition.

Portion of an example source connector configuration using topic creation rules

```
...
topic.creation.groups=compacted, highly_parallel
topic.creation.default.replication.factor=3
topic.creation.default.partitions=5

topic.creation.highly_parallel.include=hpc.*,parallel.*
topic.creation.highly_parallel.exclude=.*internal, .*metadata, .*config.*
topic.creation.highly_parallel.replication.factor=1
topic.creation.highly_paralle.partitions=1

topic.creation.compacted.include=configurations.*
topic.creation.compacted.cleanup.policy=compact
...
```

Sink Connector Configuration

This feature **does not** affect sink connectors or their configuration. Any topic creation properties added to sink connectors will be ignored and will produce a warning in the log.

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 should not break after the extensions proposed by this KIP are introduced.

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.

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.

To address cases in which the security settings need to differ for the Connect worker and its ability to create Connect's internal topics and a source connector that needs to be allowed to create new topics using the feature described in this KIP, Connect will give the ability of such specialization via the config overrides that were introduced with KIP-458. For example, if a specific source connector contains the right properties with the prefix `admin.override.` then this connector will be allowed to create new topics in cases where the Connect worker's settings would not be the appropriate. The following two examples highlight two different use cases.

Example 5: The connector is deployed with special properties that work both for producing records and creating topics:

Portion of an example source connector configuration using topic creation rules

```
...
producer.override.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="alice" \
    password="alice-secret";
...
```

Example 6: The connector is deployed with special properties that work both for producing records and creating topics:

Portion of an example source connector configuration using topic creation rules

```
...
producer.override.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="alice" \
    password="alice-secret";

admin.override.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="bob" \
    password="bob-secret";
...
```

If neither the worker properties or the connector overrides allow for creation of new topics during runtime, if this feature is enabled, an error will be logged and the task will fail.

If creating topics is not desired for security purposes, this feature should be disabled by setting `topic.creation.enable=false`. In this case, the previous behavior of assuming the topics already exist or that the broker will auto-create them when needed will be in effect.

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 by default. However, 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 an entire Connect cluster, 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.

Existing topics will also see no changes after connectors get reconfigured, since - as it's mentioned in the API description - these configuration properties apply only to new topics at the time that these properties are set.

Finally, this feature uses Kafka's Admin API methods to check for the existence of a topic and to create new topics. If the broker does not support the Admin API methods, an error will be logged and the task will fail if this feature is enabled. 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, then appropriate overrides will have to be present in the connector's configuration, and, finally, if that's not in place either, then an error will be logged and the task will fail if this feature is enabled.

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.enable` 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 `topic.creation.enable` 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.