Cloudera Streams Messaging Operator for Kubernetes 1.6.0

Upgrading Cloudera Streams Messaging Operator for Kubernetes

Date published: 2024-06-11
Date modified: 2026-01-27

https://docs.cloudera.com/

Legal Notice

Contents

Upgrading and rolling back Cloudera Streams Messaging Operator for Kubernetes

Learn about upgrading and rolling back Cloudera Streams Messaging Operator for Kubernetes.

Upgrades

Important: Before you upgrade, migrate all your ZooKeeper-based Kafka clusters to KRaft. Upgrading a ZooKeeper-based Kafka cluster to this version is not supported. For more information, see Migrating Kafka clusters from ZooKeeper to KRaft.

Upgrading Cloudera Streams Messaging Operator for Kubernetes consists of upgrading Strimzi, Kafka, and Kafka Connect.

Upgrading Strimzi involves upgrading the Strimzi Custom Resource Definitions (CRD) and upgrading the Strimzi Cluster Operator using Helm.

Upgrading Kafka and Kafka Connect involves updating your `Kafka` and `KafkaConnect` resources so that they specify the latest supported Kafka version. Upgrading Kafka and Kafka Connect is done following a Strimzi upgrade. Upgrading Kafka and Kafka Connect is:

- Strongly recommended by Cloudera after every Strimzi upgrade.
- Mandatory if you are upgrading from a maintenance version or if you are using a custom Kafka image.

Upgrading Strimzi might affect the `Kafka` and `KafkaConnect` resources in the cluster. All Kafka and Kafka Connect clusters that specify the version of the cluster but not the image are restarted when you upgrade Strimzi. This is due to the fact that the default image of all versions changes with the Strimzi upgrade, triggering a restart. This restart is safe, that is, all healthy topics with a proper replication factor and minimum ISR are kept online during the restart.

Rollbacks

A rollback involves rolling back Kafka and Kafka Connect as well as Strimzi to a previous version.

A rollback is only possible if the Kafka version you are upgrading from is still supported by the Cloudera Streams Messaging Operator for Kubernetes version you are upgrading to. If the Kafka version is not supported, upgrading is possible, but rollback is not. If you are upgrading to a version that does not support your current Kafka version and you want to have the ability to roll back, you might need to carry out multiple upgrades.

Upgrading Cloudera Streams Messaging Operator for Kubernetes

To upgrade Cloudera Streams Messaging Operator for Kubernetes, you upgrade Strimzi as well as your Kafka and Kafka Connect clusters.

Before you begin

- Ensure that your Kubernetes environment meets requirements listed in System requirements.
- Ensure that you have access to your Cloudera credentials (username and password). Credentials are required to access the Cloudera Archive and Cloudera Docker registry where upgrade artifacts are hosted.
- If you are upgrading from a maintenance version, check that the bug fixes provided in the maintenance version are available in the newer Kafka supported by Cloudera Streams Messaging Operator for Kubernetes. If certain fixes are not available, be aware that upgrading will result in regressed functionality.

- If you are using a custom Kafka image, build new custom images that are based on the Kafka images shipped with the Cloudera Streams Messaging Operator for Kubernetes version you are upgrading to.

  Cloudera Streams Messaging Operator for Kubernetes ships with two Kafka images, one for the latest supported version of Kafka, and one for the previously supported version of Kafka. Build new custom images based on both.

  The custom image based on the latest supported version is required for the upgrade. The custom image based on the previously supported version is required for rollbacks.

  Note: If you have Ranger integration set up, you are using a custom image. Building new custom images is required.

- The following steps instruct you to set spec.metadataVersion during the upgrade to keep Kafka in backward compatible state during the upgrade. This is only necessary if the metadata version differs between your current and new versions.

  If there is no change in the protocol version, you also do not finalize the upgrade. This means that for these types of upgrades, a rollback is possible even after you completed the upgrade.

  You can find the Kafka protocol version in Component versions.

Procedure

1. Update the Strimzi CRDs.

   You do this by replacing the currently installed CRDs with the new version. The CRDs are published as a single YAML on the Cloudera Archive in /p/csm-operator/1.6/install.

   ```
   curl -s --user [***USERNAME***] \
     https://archive.cloudera.com/p/csm-operator/1.6.0/install/strimzi-crd
   s-0.49.1.1.6.0-b99.yaml \
   | kubectl replace --filename -
   ```

   Enter your Cloudera password when prompted.

2. If you are upgrading from a maintenance version or are using a custom Kafka image, pause reconciliation for all your Kafka clusters.

   Important: During the period when reconciliation is paused, your cluster is not managed by Strimzi. Even if a pod crashes, it is not restarted. Ensure that you prepare for this time window to avoid accidental issues.

   You pause reconciliation by setting the strimzi.io/pause-reconciliation annotation to true in the `Kafka` resource.

   ```
   kubectl annotate kafka [***KAFKA CLUSTER NAME***] /
     --namespace [***KAFKA NAMESPACE***] /
      strimzi.io/pause-reconciliation="true"
   ```

   Pausing reconciliation ensures that the Kafka cluster does not get automatically updated during the upgrade of Strimzi. Reconciliation is only paused temporarily. You re-enable it after the Strimzi upgrade. Ensure that you add the annotation to all your Kafka clusters.

3. If you are upgrading from a maintenance version, or you are using a custom Kafka image, pause reconciliation for all your Kafka Connect clusters.

   Important: During the period when reconciliation is paused, your cluster is not managed by Strimzi. Even if a pod crashes, it is not restarted. Ensure that you prepare for this time window to avoid accidental issues.

   You pause reconciliation by setting the strimzi.io/pause-reconciliation annotation to true in the `KafkaConnect` resource.

   ```
   kubectl annotate kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] /
     --namespace [***KAFKA CONNECT NAMESPACE***] /
   ```

```
    strimzi.io/pause-reconciliation="true"
```

Pausing reconciliation ensures that the Kafka Connect cluster does not get automatically updated during the upgrade of Strimzi. Reconciliation is only paused temporarily. You reenable it after the Strimzi upgrade. Ensure that you add the annotation to all your Kafka Connect clusters.

4. Log in to the Cloudera Docker registry with helm.

```
helm registry login container.repository.cloudera.com
```

Enter your Cloudera credentials when prompted.

5. Upgrade Strimzi using helm upgrade.

This step upgrades the Strimzi Cluster Operator. Under the hood, the Strimzi Cluster Operator deployment is updated by changing the image used by the Strimzi Cluster Operator.

```
helm upgrade strimzi-cluster-operator \
  --namespace [***STRIMZI CLUSTER OPERATOR NAMESPACE***] \
  --atomic \
  oci://container.repository.cloudera.com/cloudera-helm/csm-operator/str
imzi-kafka-operator \
--version 1.6.0-b99
```

- The string strimzi-cluster-operator is the Helm release name of the chart installation. This is an arbitrary, user-defined name. Replace this string if you used a different name when you installed Strimzi.
- The --atomic option makes the command wait for the upgrade to complete or roll back if the upgrade fails.

Note: Usually, you do not need to set new Helm chart properties during upgrade, the properties that you configured during installation are preserved. However, If you set even a single property using the --set options in your helm upgrade command, properties set previously might be ignored. If you decide to set properties during the upgrade, ensure that you do one of the following.

- Set all your properties (including the ones that were set during installation) during upgrade. This ensures that all required properties are set explicitly.
- Set the properties you want to update and use the --reset-then-reuse-values option. Using this option preserves previously set properties. This option is only available in the more recent versions of Helm.

6. Verify that the Strimzi upgrade is successful.

To do this, check that the Strimzi Cluster Operator deployment is in a healthy state.

```
kubectl get deployments strimzi-cluster-operator \
  --namespace [***STRIMZI CLUSTER OPERATOR NAMESPACE***]
```

7. Upgrade Kafka.

To do this, update your `Kafka` resources for each cluster.

```
kubectl edit kafka [***KAFKA CLUSTER NAME***] \
  --namespace [***KAFKA NAMESPACE***]
```

a) Set spec.kafka.version to the latest version.

b) If you are using a custom image, set your new custom image that corresponds with the new Kafka version in spec.kafka.image.

c) Set spec.metadataVersion to the old metadata version.

This is needed to keep the Kafka cluster in a backward-compatible state after an upgrade.

```
#...
kind: Kafka
spec:
  kafka:
    version: 4.1.1.1.6
    metadataVersion: 4.0-IV3
```

d) Save the changes made to the `Kafka` resource.

e) If you paused reconciliation of the Kafka cluster, remove the strimzi.io/pause-reconciliation annotation.

   This re-enables reconciliation for the cluster.

```
kubectl annotate kafka [***KAFKA CLUSTER NAME***] /
  --namespace [***KAFKA NAMESPACE***] /
  strimzi.io/pause-reconciliation-
```

8. Verify that the Kafka upgrade is successful.

Upgrade is successful once the `Kafka` resources are in a ready state.

```
kubectl get kafkas \
  --namespace [***KAFKA NAMESPACE***] \
  --watch
```

9. Upgrade Kafka Connect.

To do this, update your `KafkaConnect` resources for each cluster.

```
kubectl edit kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] \
  --namespace [***KAFKA CONNECT NAMESPACE***]
```

a) Set spec.version to the latest version.

b) If you are using a custom image, set your new custom image that corresponds with the new Kafka version in spec.image.

c) Save the changes made to the `KafkaConnect` resource.

d) If you paused reconciliation of the Kafka Connect cluster, remove the strimzi.io/pause-reconciliation annotation.

   This re-enables reconciliation for the cluster.

```
kubectl annotate kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] /
  --namespace [***KAFKA CONNECT NAMESPACE***] /
  strimzi.io/pause-reconciliation-
```

10. Verify that the Kafka Connect upgrade is successful.

Upgrade is successful once the `KafkaConnect` resources are in a ready state.

```
kubectl get kafkaconnects \
  --namespace [***KAFKA CONNECT NAMESPACE***] \
  --watch
```

11. Finalize the upgrade.

Important: A rollback is no longer possible after you complete this step.

To do this, remove the spec.metadataVersion from your `Kafka` resource.

Rolling back Cloudera Streams Messaging Operator for Kubernetes

To roll back Cloudera Streams Messaging Operator for Kubernetes to a previous version, you roll back your Kafka and Kafka Connect clusters as well as Strimzi.

Before you begin

- Rollbacks are only possible if:
  - The Kafka version you upgraded from is still supported by the Cloudera Streams Messaging Operator for Kubernetes version that you upgraded to.
  - The Kafka upgrade is not finalized. A Kafka upgrade is finalized if spec.metadataVersion for the Kafka cluster is set to the current metadata version.
- If you are using a custom Kafka image:
  - Ensure that you have a custom Kafka image that is based on the latest base image with the previous Kafka version. This image is only set and used temporarily during the rollback.

    For example, Cloudera Streams Messaging Operator for Kubernetes 1.6 ships two Kafka base images. One for Kafka 4.1.1.1.6 (latest/current) and one for Kafka 4.0.1.1.5 (previous). In order to rollback, you need a custom image based on the 4.0.1.1.5 image shipped with 1.6.

    The base image used to build the custom image must be an image shipped in the Cloudera Streams Messaging Operator for Kubernetes version you upgraded to. Even though earlier Cloudera Streams Messaging Operator for Kubernetes versions might have shipped the same Kafka version, you cannot use a base image from a previous Cloudera Streams Messaging Operator for Kubernetes release even if the Kafka version is the same.
  - Ensure that you have access to the original custom image that you used before you upgraded. You will be rolling back your clusters to use this image.
  - If you are rolling back to a maintenance version, it can happen that during the rollback your cluster temporarily runs with a Kafka image that does not have all fixes included in the maintenance version.

    This is because a rollback is done in two stages. First, you roll back Kafka and Kafka Connect, then you roll back Strimzi. After rolling back Kafka or Kafka Connect your clusters are automatically restarted and use a Kafka image that includes the previous version of Kafka. However, that image might not contain all fixes that are in the maintenance version you are rolling back to. This is only temporary, once you rollback Strimzi, your cluster will fully revert to using the initial maintenance version you upgraded from.

Procedure

1. **Roll back Kafka.**

   To do this, edit your `Kafka` resources.

   ```
   kubectl edit kafka [***KAFKA CLUSTER NAME***] \
     --namespace [***KAFKA NAMESPACE***]
   ```

   a) Set spec.kafka.version to the previous version.

   b) If you are using a custom Kafka image, set spec.kafka.image to a custom Kafka image that is based on the latest base image with the previous Kafka version.

      The image you set here is not the original custom image that was used before you completed the upgrade. Strimzi at this stage is still upgraded and is unable to manage your original image. Because of this, you must specify a custom Kafka image that is based on the latest base image with the previous Kafka version.

   c) Save the changes made to the `Kafka` resource.

   After the changes are saved, the Kafka cluster is restarted in a rolling fashion.
2. **Roll back Kafka Connect.**

   To do this, edit your `KafkaConnect` resources.

   ```
   kubectl edit kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] \
   ```

```
  --namespace [***KAFKA CONNECT NAMESPACE***]
```

a) Set spec.version to the previous version.
b) If you are using a custom Kafka image, set spec.image to a custom Kafka image that is based on the latest base image with the previous Kafka version.

   The image you set here is not the original custom image that was used before you completed the upgrade. Strimzi at this stage is still upgraded and is unable to manage your original image. Because of this, you must specify a custom Kafka image that is based on the latest base image with the previous Kafka version
c) Save the changes made to the `KafkaConnect` resource.

After the changes are saved, the Kafka Connect cluster is restarted in a rolling fashion.

3. If you are using a custom Kafka image, pause reconciliation of the Kafka clusters.

To do this, add the strimzi.io/pause-reconciliation="true" annotation to your `Kafka` resources.

```
kubectl annotate kafka [***KAFKA CLUSTER NAME***] /
  --namespace [***KAFKA NAMESPACE***] /
   strimzi.io/pause-reconciliation="true"
```

4. If you are using a custom Kafka image, pause reconciliation of the Kafka Connect clusters.

To do this, add the strimzi.io/pause-reconciliation="true" annotation to your `KafkaConnect` resources.

```
kubectl annotate kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] /
  --namespace [***KAFKA CONNECT NAMESPACE***] /
   strimzi.io/pause-reconciliation="true"
```

5. Roll back Strimzi with helm.

```
helm rollback strimzi-cluster-operator \
  --namespace [***STRIMZI CLUSTER OPERATOR NAMESPACE***]
```

Note: If you did not pause reconciliation, both Kafka and Kafka Connect clusters managed by the Strimzi Cluster Operator are restarted in a rolling fashion when you roll back Strimzi.

6. Verify that the Strimzi rollback is successful.

To do this, check that the Strimzi Cluster Operator deployment is in a healthy state.

```
kubectl get deployments strimzi-cluster-operator \
  --namespace [***STRIMZI CLUSTER OPERATOR NAMESPACE***]
```

7. If you are using a custom Kafka image, set spec.kafka.image in your `Kafka` resources to the original custom Kafka image that was in use before you completed the upgrade.

8. If you are using a custom Kafka image, set spec.image in your `KafkaConnect` resources to the original custom Kafka image that was in use before you completed the upgrade.

9. If you paused reconciliation for Kafka clusters, remove the strimzi.io/pause-reconciliation annotation from your `Kafka` resources.

This re-enables reconciliation for the cluster.

```
kubectl annotate kafka [***KAFKA CLUSTER NAME***] /
  --namespace [***KAFKA NAMESPACE***] /
   strimzi.io/pause-reconciliation-
```

10. If you paused reconciliation for Kafka Connect clusters, remove the strimzi.io/pause-reconciliation annotation from your `KafkaConnect` resources.

This re-enables reconciliation for the cluster.

```
kubectl annotate kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] /
  --namespace [***KAFKA CONNECT NAMESPACE***] /
```

```
    strimzi.io/pause-reconciliation-
```

11. Verify that both Kafka and Kafka Connect rollbacks are successful.

    Rollback is successful once your `Kafka` and `KafkaConnect` resources are in a ready state.

    ```
    kubectl get kafkas \
      --namespace [***KAFKA NAMESPACE***] \
      --watch
    ```

    ```
    kubectl get kafkaconnects \
      --namespace [***KAFKA CONNECT NAMESPACE***] \
      --watch
    ```

Upgrading and rolling back Cloudera Surveyor for Apache Kafka

Learn about upgrading and rolling back Cloudera Surveyor. Upgrading and rolling back Cloudera Surveyor is done with helm commands. Use helm upgrade to upgrade to a new release, and helm rollback to roll back to a previous release. Both operations are seamless and require no downtime, ensuring uninterrupted service.

Upgrading Cloudera Surveyor

You upgrade Cloudera Surveyor using helm upgrade. The upgrade is a two-step process. First, you upgrade the Cloudera Surveyor server, afterward, you upgrade the Cloudera Surveyor UI. You also use helm history and take note of revisions before and during the upgrade.

Before you begin

During the upgrade, when the server is upgraded but the UI is not, you might encounter minor UI issues. For example, the configuration of some topics might fail to load. This is temporary, all data is loaded correctly once the upgrade finishes.

Procedure

1. List available Helm revisions and take note of the currently deployed revision.

    ```
    helm history cloudera-surveyor \
      --namespace [***SURVEYOR NAMESPACE***]
    ```

    Taking note of the revision now makes future rollbacks easier, as it can be difficult to identify the correct revision later. The rollback steps refer to this revision as *[***ORIGINAL REVISION***]*.

2. Upgrade the Cloudera Surveyor server.

    ```
    helm upgrade cloudera-surveyor \
      --namespace [***SURVEYOR NAMESPACE***] \
      --atomic \
      --reset-then-reuse-values \
      --set=image.tag=0.1.0.1.6.0-b99 \
      --set=image.uiTag=0.1.0.1.5.0-b123 \
      oci://container.repository.cloudera.com/cloudera-helm/csm-operator/surv
    eyor \
    ```

```
  --version 1.6.0-b99
```

- The string cloudera-surveyor is the Helm release name of the chart installation. This is an arbitrary, user-defined name. Replace this string if you used a different name when you installed Cloudera Surveyor.
- The --atomic option makes the command wait for the upgrade to complete or roll back if the upgrade fails.
- The --reset-then-reuse-values option is used to ensure that previously set configurations are preserved. This option is required because the upgrade command also uses --set options.
- The --set options specify the exact image tags to use for the server and UI. In this step, only the server image tag (image.tag=0.1.0.1.6.0-b99), is updated. The UI image tag (image.uiTag=0.1.0.1.5.0-b123) remains unchanged. This ensures that only the server is upgraded.

3. Take note of the Helm revision.

The revision is printed in the output of the helm upgrade command. The revision is needed in case you want to roll back the upgrade. The rollback steps refer to this revision as *[***SERVER UPGRADE REVISION***]*.

4. Upgrade the Cloudera Surveyor UI.

```
helm upgrade cloudera-surveyor \
  --namespace [***SURVEYOR NAMESPACE***] \
  --atomic \
  --reset-then-reuse-values \
  --set=image.tag=0.1.0.1.6.0-b99 \
  --set=image.uiTag=0.1.0.1.6.0-b99 \
  oci://container.repository.cloudera.com/cloudera-helm/csm-operator/surv
eyor \
  --version 1.6.0-b99
```

In this step, both the server (image.tag=0.1.0.1.6.0-b99) and UI (image.uiTag=0.1.0.1.6.0-b99) image tags are updated, completing the upgrade for the entire application.

5. Verify that the upgrade is successful.

To do this, check that the Cloudera Surveyor `Deployment` is in a healthy state.

```
kubectl get deployments cloudera-surveyor \
  --namespace [***SURVEYOR NAMESPACE***]
```

In addition, you can check the image version that the `Pods` are using.

```
kubectl get pods \
  --namespace=kafka \
  --output custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*
].image"
```

On successful upgrade, the `Pods` will use the new version of the Cloudera Surveyor image.

```
NAME                         IMAGE
cloudera-surveyor-558595d999-dt2dj docker-private.infra.cloudera.com/clo
udera/surveyor:0.1.0.1.6.0-b99
```

What to do next

Reload the browser page where you access the UI to ensure that the latest version of the UI is displayed. The browser may continue to show a previous version of the UI until you refresh.

Rolling back Cloudera Surveyor

You roll back Cloudera Surveyor using helm rollback. A rollback involves listing and identifying the upgrade revisions that you want to roll back to. Afterward, you complete a two-step rollback process where you roll back the Cloudera Surveyor UI and the Cloudera Surveyor server.

Before you begin

- During the rollback, when the server is still upgraded but the UI is already rolled back, you might encounter minor UI issues. For example, the configuration of some topics might fail to load. This is temporary, all data is loaded correctly once the upgrade finishes.
- If you took note of your revisions during the upgrade process, skip steps 1 on page 12 and 2 on page 12.

Procedure

1. List available Helm revisions.

```
helm history cloudera-surveyor \
  --namespace [***SURVEYOR NAMESPACE***]
```

2. Identify the revision that upgraded the Cloudera Surveyor server as well as the original revision you upgraded from.

For example, if you just completed an upgrade, you need the previous two revisions. These are revisions 1 and 2 in the following example.

```
REVISION  #...  STATUS      CHART                 DESCRIPTION
1         #...  superseded  surveyor-1.5.0-b123   Install complete
2         #...  superseded  surveyor-1.6.0-b99   Upgrade complete
3         #...  deployed    surveyor-1.6.0-b99   Upgrade complete
```

You can also identify revisions by looking at the chart version.

- The revision that upgraded the server is the first revision that has the latest chart version. This is revision 2 in the example.
- The original revision you upgraded from is the last revision that has the old chart version. This is revision 1 in the example.

3. Roll back the Cloudera Surveyor UI.

```
helm rollback cloudera-surveyor [***SERVER UPGRADE REVISION***] \
  --namespace [***SURVEYOR NAMESPACE***]
```

Replace *[***SERVER UPGRADE REVISION***]* with the revision that upgraded the Cloudera Surveyor server.

4. Roll back the Cloudera Surveyor server.

```
helm rollback cloudera-surveyor [***ORIGINAL REVISION***] \
  --namespace [***SURVEYOR NAMESPACE***]
```

Replace *[***ORIGINAL REVISION***]* with the revision that you originally upgraded from.

5. Verify that the rollback is successful.

To do this, check that the Cloudera Surveyor `Deployment` is in a healthy state.

```
kubectl get deployments cloudera-surveyor \
  --namespace [***SURVEYOR NAMESPACE***]
```

In addition, you can check the image version that the Cloudera Surveyor `Pods` are using.

```
kubectl get pods \
  --namespace [***SURVEYOR NAMESPACE***] \
  --output custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].
image"
```

On successful upgrade, the `Pods` will use the old version of the Cloudera Surveyor image.

```
NAME                          IMAGE
cloudera-surveyor-558595d999-dt2dj docker-private.infra.cloudera.com/clo
udera/surveyor:0.1.0.1.5.0-b123
```

What to do next

Reload the browser page where you access the UI to ensure that the latest version of the UI is displayed. The browser may continue to show a previous version of the UI until you refresh.