

Cloudera Streams Messaging Operator 1.2.0

Upgrade

Date published: 2024-06-11

Date modified: 2024-12-02

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Upgrading Cloudera Streams Messaging - Kubernetes Operator.....	4
--	----------

Upgrading Cloudera Streams Messaging - Kubernetes Operator

To upgrade Cloudera Streams Messaging - Kubernetes Operator, you upgrade Strimzi, Kafka, and Kafka Connect in your cluster.

About this task

Upgrading Strimzi consists of upgrading the Strimzi CRDs to the new version and upgrading the Strimzi Cluster Operator using Helm commands. If you are upgrading from a maintenance version, you also need to temporarily set the image of the Kafka and Kafka Connect clusters to the maintenance version in your resources. This is necessary because without the explicit image, the Strimzi upgrade will automatically update the image used by the Kafka and Kafka Connect clusters, which might not contain all bug fixes provided in the maintenance version.

Upgrading the cluster operator may 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 will be restarted during the cluster operator upgrade. 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.

Upgrading Kafka and Kafka Connect involves updating your `Kafka` and `KafkaConnect` resources for each cluster to the latest supported version after 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.

This procedure upgrades both Kafka and Kafka Connect clusters in a rolling upgrade. The upgrade is safe, all healthy topics with a proper replication factor and minimum ISR are kept online during the upgrade.



Important: A rollback is only possible if the Kafka version you are upgrading from is still supported by the Cloudera Streams Messaging - Kubernetes Operator version you are upgrading to. If the Kafka version is not supported, upgrading is possible, but a 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.

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 - Kubernetes Operator. If certain fixes are not available, be aware that upgrading will result in regressed functionality.
- If you built a custom Kafka image based on the Kafka image shipped in Cloudera Streams Messaging - Kubernetes Operator, build a new Kafka image that is based on the Kafka image shipped in the Cloudera Streams Messaging - Kubernetes Operator version you are upgrading to.
- The following steps instruct you to set `inter.broker.protocol.version` during the upgrade to keep Kafka in backward compatible state during the upgrade. This is only necessary if the protocol 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/1.2/install`.

```
curl -s --user [***USERNAME***] \
  https://archive.cloudera.com/p/csm-operator/1.2.0/install/strimzi-crd
s-0.43.0.1.2.0-b54.yaml \
| kubectl replace --filename -
```

Enter your Cloudera password when prompted.

2. If you are upgrading from a maintenance version, explicitly set the image of the Kafka cluster to the maintenance version in your Kafka resources.

These steps ensure that the Kafka clusters keep the exact maintenance image during the Strimzi upgrade. This is a temporary setup, and you need to change it after the Strimzi upgrade. Repeat these steps for each Kafka cluster.

a) Extract the currently used ZooKeeper image.

```
kubectl get pod --namespace [***KAFKA CLUSTER NAMESPACE***] \
--selector strimzi.io/component-type=zookeeper,strimzi.io/clust
er=[***KAFKA CLUSTER NAME***] \
--output=jsonpath="{@[ 'items' ][0].spec.containers[0].image}{'\n'}"
```

b) Extract the currently used Kafka image.

```
kubectl get pod --namespace [***KAFKA CLUSTER NAMESPACE***] \
--selector strimzi.io/component-type=kafka,strimzi.io/clust
er=[***KAFKA CLUSTER NAME***] \
--output=jsonpath="{@[ 'items' ][0].spec.containers[0].image}{'\n'}"
```

c) Update the `spec.zookeeper.image` and the `spec.kafka.image` properties in the Kafka resource.

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

3. If you are upgrading from a maintenance version, explicitly set the image of the Kafka Connect cluster to the maintenance version in your KafkaConnect resources.

These steps ensure that the Kafka Connect clusters keep the exact maintenance image during the Strimzi upgrade. This is a temporary setup, and you need to change it after the Strimzi upgrade. Repeat these steps for each Kafka Connect cluster.

a) Check if the image is set explicitly.

```
kubectl get kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] \
--namespace [***KAFKA CONNECT CLUSTER NAMESPACE***] \
--output=jsonpath="{.spec.image}{'\n'}"
```

b) If previous command returns empty, get the version of the cluster:

```
kubectl get kafkaconnect [***KAFKA CONNECT CLUSTER NAME***] \
--namespace [***KAFKA CONNECT CLUSTER NAMESPACE***] \
--output=jsonpath="{.spec.version}{'\n'}"
```

c) Find the default image for the version.

```
kubectl get deployments strimzi-cluster-operator --namespace [***CLUSTER
OPERATOR NAMESPACE***] \
--output 'jsonpath={.spec.template.spec.containers[0].env[?(@.name=
"STRIMZI_KAFKA_IMAGES")].value}' \
| grep [***KAFKA CONNECT VERSION***] | cut -d'=' -f2
```


- d) Update the `spec.image` property in the `KafkaConnect` resource.

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

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.

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/strimzi-kafka-operator \
  --version 1.2.0-b54
```

- 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.

- a) Ensure that the Strimzi Cluster Operator is in a healthy state.

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

- b) If needed, roll back this procedure with `helm rollback`.

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

7. Upgrade Kafka.

This is done by updating your Kafka resources for each cluster.

```
kubectl edit kafka [***KAFKA CLUSTER NAME***] \
```



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

- a) Set the spec.kafka.version to the latest version.
- b) Set or remove image locations.
 - If you are using a custom image, set the spec.zookeeper.image and spec.kafka.image properties to the location of your new custom image.
 - If you are not using a custom image and are upgrading from a maintenance version, remove the spec.zookeeper.image and spec.kafka.image properties.
- c) Add the old inter.broker.protocol.version in spec.kafka.config, for example:

```
#...
kind: Kafka
spec:
  kafka:
    version: 3.8.0.1.2
    config:
      inter.broker.protocol.version: "3.7"
```

Specifying inter.broker.protocol.version keeps Kafka in a backward-compatible state following the upgrade.

- d) Save the changes made to the Kafka resource.
8. Verify that Kafka upgrades are successful.
- a) Wait for the Kafka cluster pods to become ready.

```
kubectl get pod --namespace [***KAFKA NAMESPACE***] \
  --selector strimzi.io/cluster=[***KAFKA CLUSTER NAME***] \
  --watch
```

- b) If needed, roll back the Kafka upgrade by editing the Kafka resource.
 - Set spec.kafka.version to the previous version.
 - Set spec.zookeeper.image and spec.kafka.image to the previously used images.
 - Save the changes made to the Kafka resource.
9. Upgrade Kafka Connect.

This is done by updating your KafkaConnect resources for each cluster.

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

- a) Set the spec.version to the latest version.
 - b) Set or remove image locations.
 - If you are using a custom image, set the spec.image property to the location of your new custom image.
 - If you are not using a custom image and are upgrading from a maintenance version, remove the spec.image property.
 - c) Save the changes made to the KafkaConnect resource.
10. Verify that Kafka Connect upgrade is successful.
- a) Wait for the Kafka Connect cluster pods to become ready.

```
kubectl get pod --namespace [***KAFKA CONNECT NAMESPACE***] \
  --selector strimzi.io/cluster=[***KAFKA CONNECT CLUSTER NAME***] \
  --watch
```

- b) If needed, roll back the Kafka Connect upgrade by editing the KafkaConnect resource.
 - Set the spec.version to the previous version.
 - Set spec.image to the previous image.
 - Save the changes made to the KafkaConnect resource.

11. Finalize the Kafka upgrade by removing the old `inter.broker.protocol.version`.

After this step, rollback is no longer possible.