Cloudera Streams Messaging - Kubernetes Operator 1.3.0

Release Notes

Date published: 2024-06-11
Date modified: 2025-02-28

https://docs.cloudera.com/

Legal Notice

Contents

Release notes....4
What's New....4
Fixed Issues....5
Known Issues....5
Unsupported features....6
Deprecations and removals....6

Component versions....6

System requirements....7

Kafka Connect plugins....8

Release notes

Learn about the new features, improvements, known and fixed issues, limitations, unsupported features, as well as deprecations and removals in this release of Cloudera Streams Messaging - Kubernetes Operator.

What's New

Learn about the new features and notable changes in this release.

Rebase to Strimzi 0.45.0 and Kafka 3.9.

This release of Cloudera Streams Messaging - Kubernetes Operator is based on Strimzi 0.45.0 and Kafka 3.9.

See the following upstream resources for more information on these versions.

- Strimzi 0.44.0 Release notes
- Strimzi 0.45.0 Release notes
- Kafka 3.9.0 Release notes
- Kafka 3.9.0 Notable changes

Upstream highlights

The following is a list of highlighted changes included in the upstream version of Strimzi, Kafka, and other components. For a full list of upstream changes, see the release notes and notable changes above.

- strimzi-kafka-operator #10563: Add mechanism to manage offsets through `KafkaConnector` and `KafkaMirrorMaker2` resources

  You can now manage connector offsets directly by configuring your `KafkaConnector` resources. Cloudera recommends that you use this feature over the Kafka Connect REST API to manage connector offsets. The recommended method for managing replication offsets when replicating data with Kafka Connect-based replication is also changed. For more information, see Managing connector offsets and Configuring data replication offsets.

  Note: Cloudera Streams Messaging - Kubernetes Operator does not support the `KafkaMirrorMaker2` resource. While the feature is available for this resource, the use of the resource is not recommended or supported.
- strimzi-kafka-operator #10583: Auto-rebalancing on scaling the cluster down/up

  You can now enable auto-rebalancing for Kafka clusters. If auto-rebalancing is enabled, the Strimzi Cluster Operator automatically initiates a rebalance with Cruise Control when you scale the Kafka cluster. Cloudera recommends that you enable this feature as it makes scaling easier and faster. For more information, see Scaling brokers.

Supported Kubernetes version update

Starting with this release, Cloudera Streams Messaging - Kubernetes Operator supports the following Kubernetes versions.

- Kubernetes 1.25 or later:
  - OpenShift 4.12 or later
  - RKE2 (Rancher Kubernetes Engine 2) 1.25 or later

KRaft

KRaft (Kafka Raft) is generally available. You can now deploy Kafka clusters that use KRaft instead of ZooKeeper for metadata management. Additionally, you can migrate existing ZooKeeper-based Kafka clusters to use KRaft.

With the addition of KRaft, ZooKeeper is deprecated. Deploying new or using existing Kafka clusters running in ZooKeeper mode is deprecated. Additionally, ZooKeeper will be removed in a future release. When deploying new Kafka clusters, deploy them in KRaft mode. Cloudera encourages you to migrate existing clusters to KRaft.

- For cluster deployment instructions, see Deploying a Kafka cluster.
- For migration instructions, see Migrating Kafka clusters from ZooKeeper to KRaft.

Fixed Issues

Learn what issues have been fixed since the previous release.

CSMDS-933: The log-dir-describe.txt and topic-describe.txt files generated by report.sh are empty

The report.sh tool now correctly collects log directory and topic information even if you run multiple concurrent instances of the tool, or run the tool with a high parallelization factor. Both log-dir-describe.txt and topics-describe.txt are correctly populated with information.

Known Issues

Learn about the known issues in this release.

CSMDS-334: ZooKeeper pods are running but Kafka pods are not created

Under certain circumstances, ZooKeeper pods might not be able to form a quorum. In a case like this, the creation of the Kafka cluster gets stuck in a state where ZooKeeper pods are running, but Kafka pods are not created.

If you encounter this issue, at least one of the ZooKeeper pods logs a WARN entry similar to the following:

```
2024-02-23 18:45:00,311 WARN Unexpected exception (org.apache.zo
okeeper.server.quorum.QuorumPeer) [QuorumPeer[myid=3](plain=127.
0.0.1:12181)(secure=[0:0:0:0:0:0:0:0]:2181)]
java.lang.InterruptedException: Timeout while waiting for epoch
 from quorum
 at org.apache.zookeeper.server.quorum.Leader.getEpochToPropose
(Leader.java:1443)
 at org.apache.zookeeper.server.quorum.Leader.lead(Leader.java:60
6)
 at org.apache.zookeeper.server.quorum.QuorumPeer.run(QuorumPeer.
java:1552)
```

This is caused by a race condition issue in ZooKeeper. ZooKeeper is unable to recover from this state automatically.

Delete the ZooKeeper pods that are unable to form a quorum.

```
kubectl delete pod [***ZOOKEEPER POD***] -n [***NAMESPACE***]
```

The Strimzi Cluster Operator automatically recreates the ZooKeeper pods that are deleted. The newly created ZooKeeper pods are less likely to encounter the issue.

CSMDS-953: Kafka and ZooKeeper might experience downtime during upgrades

Under certain circumstances, ZooKeeper pods might not be able to form a quorum during an upgrade. This results in both ZooKeeper and Kafka becoming unavailable for several minutes during an upgrade.

After a certain amount of time, failed ZooKeeper pods are automatically recreated by the Strimzi Cluster Operator, and the upgrade succeeds.

If you encounter this issue, at least one of the ZooKeeper pods will log the following error:

```
java.net.BindException: Cannot assign requested address
```

This issue affects ZooKeeper-based deployments only.

Unsupported features

Learn what features are unsupported in this release.

The following Strimzi features are unsupported in Cloudera Streams Messaging - Kubernetes Operator:

- Kafka MirrorMaker
- Kafka MirrorMaker 2
- Kafka Bridge
- Kafka cluster creation without using `KafkaNodePool` resources

Deprecations and removals

Learn what is deprecated or removed in this release.

Deprecations

ZooKeeper

ZooKeeper is deprecated. Deploying new or using existing Kafka clusters running in ZooKeeper mode is deprecated. ZooKeeper will be removed in a future release. When deploying new Kafka clusters, deploy them in KRaft mode. Cloudera encourages you to migrate existing clusters to KRaft.

- For cluster deployment instructions, see Deploying a Kafka cluster.
- For migration instructions, see Migrating Kafka clusters from ZooKeeper to KRaft.

Removals

There are no removals in this release.

Component versions

A list of components and their versions shipped in this release of Cloudera Streams Messaging - Kubernetes Operator.

Table 1: Cloudera Streams Messaging - Kubernetes Operator component versions

Component	Version
Cruise Control	2.5.141.1.3.0-b52
Kafka	3.9.0.1.3.0-b52
Strimzi	0.45.0.1.3.0-b52
ZooKeeper	3.8.1.7.3.1.0-197

Supported Kafka versions

Cloudera Streams Messaging - Kubernetes Operator supports the following Kafka versions:

Table 2: Supported Kafka versions

Version	Component/Resource	Kafka Protocol version	Metadata version
3.9.0.1.3 (latest and default)	• Kafka – `Kafka` resources • Kafka Connect – `KafkaConnect` resources	3.9	3.9-IV0
3.8.0.1.2	• Kafka – `Kafka` resources • Kafka Connect – `KafkaConnect` resources	3.8	N/A

Kafka versions shipped in Cloudera Streams Messaging - Kubernetes Operator are specific to Cloudera. You specify them in the spec.version property of cluster resources like `Kafka` and `KafkaConnect` resources.

The latest version is the current and latest supported version. This version is used by default to deploy clusters if an explicit version is not provided in your resource configuration. This version is also the version recommended by Cloudera. All other versions listed here are Kafka versions shipped in previous releases of Cloudera Streams Messaging - Kubernetes Operator that are also supported.

The Kafka version is made up of two parts. The first three digits specify the Apache Kafka version. The digits following the Apache Kafka version specify the major and minor version of Cloudera Streams Messaging - Kubernetes Operator. When deploying a cluster, you must use the versions listed here. Specifying upstream versions is not supported.

The Kafka protocol and metadata version is relevant for upgrades. Protocol version is relevant for ZooKeeper-based clusters, while metadata version is relevant for KRaft-based clusters. Depending on your specific upgrade path, explicitly setting the protocol or metadata version might be necessary during an upgrade.

System requirements

Cloudera Streams Messaging - Kubernetes Operator requires a Kubernetes cluster environment that meets the following system requirements and prerequisites. Ensure that you meet these requirements, otherwise, you will not be able to install and use Cloudera Streams Messaging - Kubernetes Operator or its components.

- A Kubernetes 1.25 or later cluster.
  - OpenShift 4.12 or later.
  - Any Cloud Native Computing Foundation (CNCF) certified Kubernetes distribution. For more information, see cncf.io.
- Administrative rights on the Kubernetes cluster.
- Access to kubectl or oc. These command line tools must be configured to connect to your running cluster.
- Access to helm.
- Log collection is enabled for the Kubernetes cluster.

  Cloudera requires that the logs of Cloudera Streams Messaging - Kubernetes Operator components are stored long term for diagnostic and supportability purposes. Collect logs using the log collector feature of your Kubernetes platform. Cloudera recommends at least one week of retention time for the collected logs.
- A persistent storage class configured on the Kubernetes cluster that satisfies the durability and low-latency requirements for operating Kafka.

  The ideal storage class configuration can vary per environment and use-case and is determined by the Kubernetes platform where Cloudera Streams Messaging - Kubernetes Operator is deployed.

  Additionally, for Kafka brokers, Cloudera recommends a `StorageClass` that has volume expansion enabled (allowvolumeexpansion set to true).
- A Prometheus installation running in the same Kubernetes cluster where you install Cloudera Streams Messaging - Kubernetes Operator is recommended. Prometheus is used for collecting and monitoring Kafka and Strimzi metrics.

Kafka Connect plugins

Learn what Kafka Connect plugins are shipped with and supported in Cloudera Streams Messaging - Kubernetes Operator.

Connectors

Cloudera Streams Messaging - Kubernetes Operator ships and supports all Kafka Connect connectors included in Apache Kafka.

The full list is as follows.

- `org.apache.kafka.connect.mirror.MirrorCheckpointConnector`
- `org.apache.kafka.connect.mirror.MirrorSourceConnector`
- `org.apache.kafka.connect.mirror.MirrorHeartBeatConnector`
- `org.apache.kafka.connect.file.FileStreamSourceConnector`
- `org.apache.kafka.connect.file.FileStreamSinkConnector`

Note:

Although both `FileStreamSourceConnector` and `FileStreamSinkConnector` are shipped with Cloudera Streams Messaging - Kubernetes Operator, neither connector is installed, and you cannot deploy them by default. To deploy instances of these connectors, you must first install them as any other third-party connector. Cloudera also does not recommend that you use these connectors in production.

Single Message Transforms plugins (transformations and predicates)

Single Message Transforms (SMT) plugins (transformations and predicates) are deployed on top of Kafka Connect connectors. They enable you to apply message transformations and filtering on a single message basis. Cloudera Streams Messaging - Kubernetes Operator ships and supports all transformation and predicates plugins included in Apache Kafka as well as the `ConvertFromBytes` and `ConvertToBytes` plugins, which are Cloudera specific plugins.

The full list is as follows.

Transformations

- `com.cloudera.dim.kafka.connect.transforms.ConvertFromBytes`
- `com.cloudera.dim.kafka.connect.transforms.ConvertToBytes`
- `org.apache.kafka.connect.transforms.Cast`
- `org.apache.kafka.connect.transforms.DropHeaders`
- `org.apache.kafka.connect.transforms.ExtractField`
- `org.apache.kafka.connect.transforms.Filter`
- `org.apache.kafka.connect.transforms.Flatten`
- `org.apache.kafka.connect.transforms.HeaderFrom`
- `org.apache.kafka.connect.transforms.HoistField`
- `org.apache.kafka.connect.transforms.InsertField`
- `org.apache.kafka.connect.transforms.InsertHeader`
- `org.apache.kafka.connect.transforms.MaskField`
- `org.apache.kafka.connect.transforms.RegexRouter`
- `org.apache.kafka.connect.transforms.ReplaceField`
- `org.apache.kafka.connect.transforms.SetSchemaMetadata`
- `org.apache.kafka.connect.transforms.TimestampConverter`
- `org.apache.kafka.connect.transforms.TimestampRouter`
- `org.apache.kafka.connect.transforms.ValueToKey`

Predicates

- `org.apache.kafka.connect.transforms.predicates.HasHeaderKey`
- `org.apache.kafka.connect.transforms.predicates.RecordIsTombstone`
- `org.apache.kafka.connect.transforms.predicates.TopicNameMatches`

Converters

Converters can be used to transform Kafka record keys and values between bytes and a specific format. In addition to the `JsonConverter`, there are converters for most often used primitive types as well.

The full list is as follows.

- `org.apache.kafka.connect.json.JsonConverter`
- `org.apache.kafka.connect.converters.ByteArrayConverter`
- `org.apache.kafka.connect.converters.BooleanConverter`
- `org.apache.kafka.connect.converters.DoubleConverter`
- `org.apache.kafka.connect.converters.FloatConverter`
- `org.apache.kafka.connect.converters.IntegerConverter`
- `org.apache.kafka.connect.converters.LongConverter`
- `org.apache.kafka.connect.converters.ShortConverter`
- `org.apache.kafka.connect.storage.StringConverter`

Header converters

Header converters can be used to transform Kafka record headers between bytes and a specific format. Cloudera Streams Messaging - Kubernetes Operator and Kafka includes a single dedicated header converter, which is the `org.apache.kafka.connect.storage.SimpleHeaderConverter`.

The `SimpleHeaderConverter` is the default header converter and is adequate for the majority of use cases. In case your headers are of a specific format, like JSON, you can use any other converter listed in the Converters on page 9 section as a header converter as well.

Replication policies

A replication policy defines the basic rules of how topics are replicated from source to target clusters when using Kafka Connect-based replication to replicate Kafka data between Kafka clusters.

The full list is as follows.

- `org.apache.kafka.connect.mirror.DefaultReplicationPolicy`
- `org.apache.kafka.connect.mirror.IdentityReplicationPolicy`

Related Information

Installing Kafka Connect connector plugins
ConvertFromBytes
ConvertToBytes
Transformations | Kafka
Predicates | Kafka