

CSA Operator Management

Date published: 2024-06-15

Date modified: 2024-11-15



Legal Notice

© Cloudera Inc. 2025. 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

Monitoring and diagnostics.....	4
Log collection.....	5
Removing resources.....	5
Operator configuration.....	6
Namespace management.....	7
Updating Cloudera license.....	10

Monitoring and diagnostics

Learn about collecting diagnostics information, the diagnostic tool shipped with Cloudera Streaming Analytics - Kubernetes Operator, as well as a number of useful `kubectl` commands that you can use to gather diagnostic information.

In addition to the built-in health endpoint of the Flink Operator and using the generic `kubectl` command, Cloudera provides a separate command line tool that you can use to capture diagnostic information about your Cloudera Streaming Analytics - Kubernetes Operator installation. You can use these tools when contacting Cloudera support, or when troubleshooting issues.

Diagnostic bundle

The diagnostic tool is a Python package that collects all relevant resources and logs managed by the Cloudera Streaming Analytics - Kubernetes Operator and connects to the REST API of running Flink clusters to fetch additional metrics. It generates a zip file that can be shared with Cloudera support or examined for troubleshooting.

By default, the diagnostic tool is not downloaded, deployed, or installed when you install Cloudera Streaming Analytics - Kubernetes Operator and its components. To use it, download the Python package located in the `/csa-operator/1.0/tools/` directory on Cloudera Archive, and use the following steps to install and create the diagnostic bundle:

1. Create a Python virtual environment.

```
mkdir venv
python3 -m venv venv
cd venv
source bin/activate
```

2. Install the Cloudera Streaming Analytics diagnostic tool with `pip` install.

```
pip install ../csaop-diagnostic-1.0.0.tar.gz
```

3. Run the diagnostic tool.

```
csaop-generate-bundle
```

The following optional arguments can be provided to the diagnostic tool:

- a. By default, the diagnostic tool generates the zip file in the current working directory, but you can provide the path of a custom directory using the `-o [OUTPUT_DIR]` argument.

The path to the generated zip file is displayed when the diagnostic tool is successfully run.

Pod status with `kubectl`

You can check the status of the pods after applying a change to the deployment configuration using `kubectl describe`:

```
kubectl describe --namespace [***NAMESPACE***]
```

Operator log with `kubectl`

The Flink Operator log contains useful information about the tasks that the operator performs and details for failed operations. You can check the Flink Operator logs with `kubectl logs`:

```
kubectl logs [***FLINK_OPERATOR_POD***] --namespace [***NAMESPACE***]
```

Health endpoint

The Flink Operator provides a built-in health endpoint that serves as the information source for Kubernetes liveness and startup probes. The health probes are enabled by default in the Helm chart as shown in the following example:

```
operatorHealth:
  port: 8085
  livenessProbe:
    periodSeconds: 10
    initialDelaySeconds: 30
  startupProbe:
    failureThreshold: 30
    periodSeconds: 10
```

The health endpoint catches startup and informer errors that are exposed by the Java Operator SDK (JOSDK) framework. By default, if one of the watched namespaces becomes inaccessible, the health endpoint will report an error and the Flink Operator restarts.

If the Flink Operator needs to be running, even if some namespaces are not accessible, you can use the `kubernetes.operator.startup.stop-on-informer-error` configuration and set it to `false` to disable the automatic restart behavior. This way the Flink Operator will start even if some namespaces cannot be watched.

Log collection

Cloudera requires that the logs of the operator components are stored long term for diagnostic and supportability purposes. Learn about the settings for platform level log collection recommended by Cloudera.

Logs can be collected using the log collector feature of the specific Kubernetes platform. Ensuring that log collection is correctly set up is your responsibility. Cloudera recommends at least one week of retention time for the collected logs.

Using `kubectl logs` is not sufficient in some cases. This is because pods are created and destroyed dynamically by operator applications. The logs of destroyed pods are deleted, which makes them inaccessible. Log collection can ensure that the logs of already deleted pods are retained.

The following collects the recommended and required logging practices for specific Kubernetes platforms.

OpenShift

Latest OpenShift versions support the Vector log collector. Log collection and forwarding can be configured using a `ClusterLogging` resource.

Ensure the following if you are on OpenShift:

- The `ClusterLogging` resource includes all namespaces and pods used by the operators.
- Use a log sink that supports time-based retention. The `ClusterLogging` resource supports a number of log sinks. Cloudera recommends using a sink that supports time-based retention to limit storage costs. Additionally, the selected sink should allow easy access to the collected logs when a diagnostic investigation requires them.

Removing resources

Removing the Flink Operator and its resources before installing a newer version of the Cloudera Streaming Analytics - Kubernetes Operator.

About this task

Before installing a newer version of the Cloudera Streaming Analytics - Kubernetes Operator, you need to ensure that the Flink Operator and its resources are deleted from your namespace(s). Simply uninstalling the Cloudera Streaming Analytics - Kubernetes Operator does not mean that all of the resources are removed from the cluster.

Procedure

1. Remove the Flink Operator using the following command:

```
helm uninstall flink-operator -n flink
```

When deleting the Flink Operator, the following message indicates which resources remain in the namespace:

```
These resources were kept due to the resource policy:
[RoleBinding] flink-role-binding
[Role] flink
[ServiceAccount] flink
[PersistentVolumeClaim] postgresSQL
```

In case the PostgreSQL (or other database) persistent volume claim is not removed, the SQL project and job related data is stored in the database.

2. Remove the remaining resources using kubectl:

```
kubectl delete rolebinding flink-role-binding -n flink
kubectl delete role flink -n flink
kubectl delete serviceaccount flink -n flink
```

3. Update the Cloudera Streaming Analytics CRDs using the following commands:

```
helm template oci://container.repository.cloudera.com/cloudera-helm/csa-
operator/csa-operator --version 1.1.2-b17 --include-crds --output-dir .

kubectl replace -f csa-operator/charts/flink-kubernetes-operator/crds/fl
inkdeployments.flink.apache.org-v1.yml

kubectl replace -f csa-operator/charts/flink-kubernetes-operator/crds/fl
inksessionjobs.flink.apache.org-v1.yml
```

Operator configuration

Specify default configurations for the Flink Operator.

You can specify default configuration for the Flink Operator that is shared between the operator itself and the Flink deployments.

The configuration files are mounted externally through ConfigMaps created during the Helm chart installation. Cloudera recommends reviewing and adjusting the configurations in the values.yaml file, if applicable, before deploying the Flink Operator in production environments.

To append to the default configuration, define the flink-conf.yaml key in the flink-kubernetes-operator.defaultConfiguration section of the Helm values.yaml file:

```
defaultConfiguration:
  create: true
  # Set append to false to replace configuration files
  append: true
  flink-conf.yaml: |+
    # Flink Config Overrides
```

```
kubernetes.operator.metrics.reporter.slf4j.factory.class: org.apache.flink.metrics.slf4j.Slf4jReporterFactory
kubernetes.operator.metrics.reporter.slf4j.interval: 5 MINUTE

kubernetes.operator.reconcile.interval: 15 s
kubernetes.operator.observer.progress-check.interval: 5 s
```

The list of Flink Operator configurations can be found in *Specifying Operator Configuration*.

Dynamic Operator Configuration

The Kubernetes Operator supports dynamic config changes through the ConfigMaps of the Operator. Dynamic operator configuration is enabled by default, and can be disabled by setting `kubernetes.operator.dynamic.config.enabled` to `FALSE`. The time interval for checking dynamic config changes can be set by `kubernetes.operator.dynamic.config.check.interval`. The default value for the time interval is 5 minutes.

You can verify that the dynamic operator configuration is enabled through the `deploy/flink-kubernetes-operator` log has:

```
2022-05-28 13:08:29,222 o.a.f.k.o.c.FlinkConfigManager [INFO ] Enabled dynamic config updates, checking config changes every PT5M
```

To change configuration values dynamically the ConfigMap can be directly edited using `kubectl patch` or `kubectl edit` command.

To verify that the configuration value of `kubernetes.operator.reconcile.interval` is changed to 30 seconds, the `deploy/flink-kubernetes-operator` log should have the following information:

```
2022-05-28 13:08:30,115 o.a.f.k.o.c.FlinkConfigManager [INFO ] Updating default configuration to {kubernetes.operator.reconcile.interval=PT30S}
```



Note: Cloudera recommends setting the `kubernetes.operator.reconcile.interval` to a lower value for the changes to take effect in a shorter time.

Related Information

[Specifying Operator Configuration | Apache Flink Kubernetes Operator](#)

Namespace management

The Flink Operator is capable of watching all of the Kubernetes cluster namespaces. However, when installing the Cloudera Streaming Analytics - Kubernetes Operator, you can limit its access to a single or set number of namespaces.

The `watchNamespaces` parameter

By default, the Flink Operator is capable for watching all of the Kubernetes cluster namespaces. (The `watchNamespaces` parameter is not set.)

This means that no matter in which namespace the Flink Deployment is deployed, the Flink Operator picks it up and executes the Flink job in that namespace.

**Important:**

If the `watchNamespaces` parameter is not set, it's the responsibility of the user to deploy a Flink service account to every namespace in which they'd like to run jobs.

To add the Flink service account, follow these steps.

1. Switch to the namespace:

```
kubectl config set-context --current --namespace=[*** NAMESPACE ***]
```

2. Create the service account, role, and role binding. Use the following as an example:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: flink-kubernetes-operator
    app.kubernetes.io/version: 1.0.1
  name: flink
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  labels:
    app.kubernetes.io/name: flink-kubernetes-operator
    app.kubernetes.io/version: 1.0.1
  name: flink
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - configmaps
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  verbs:
  - '*'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
    app.kubernetes.io/name: flink-kubernetes-operator
    app.kubernetes.io/version: 1.0.1
  name: flink-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: flink
subjects:
- kind: ServiceAccount
  name: flink
```

3. (Optional) Create and run a test job in the namespace.

Limiting access to namespaces

In production environments managing access to the namespaces might be necessary. In this case, you can specify a list of namespaces the Flink Operator can watch and have access.

**Important:**

When the `watchNamespaces` parameter is set, the Helm chart automatically creates the Flink service account in every specified namespace.

The namespaces must be created before running `helm install` for the service accounts to be created.

Namespaces created **after** running `helm install` require the Flink service account added manually, as described above.

When installing the Cloudera Streaming Analytics - Kubernetes Operator, you can define the namespace configuration with `helm install`:

```
helm install csa-operator --namespace [***NAMESPACE***] \
--set 'flink-kubernetes-operator.image.imagePullSecrets[0].name=[***SECRET
NAME***]' \
--set 'ssb.sse.image.imagePullSecrets[0].name=[***SECRET NAME***]' \
--set 'ssb.sqlRunner.image.imagePullSecrets[0].name=[***SECRET NAME***]' \
--set-file clouderaLicense.fileContent=[***PATH TO LICENSE FILE***] \
--set flink-kubernetes-operator.watchNamespaces=[*** NAMESPACE(S) ***] \
oci://container.repository.cloudera.com/cloudera-helm/csa-operator/csa-op
erator --version 1.1.2-b17
```

This also means that Cloudera SQL Stream Builder does not have to be in the same namespace as the Flink Deployment. However, Cloudera SQL Stream Builder can only manage one namespace. This means that if you want to deploy it in multiple namespaces, you need to install Cloudera SQL Stream Builder in every namespace.

As an example, if you want to have two namespaces with Flink in `NAMESPACE1` and Cloudera SQL Stream Builder in `NAMESPACE2`, you need to install the Flink Kubernetes Operator in `NAMESPACE1` without Cloudera SQL Stream Builder, and install Cloudera SQL Stream Builder without the Flink Kubernetes Operator in `NAMESPACE2`.



Note: Watching multiple namespaces with the Flink Operator also means that Cloudera SQL Stream Builder does not have to be in the same namespace as the Flink Deployment. However, Cloudera SQL Stream Builder can only manage one namespace. This means that if you want to deploy it in multiple namespaces, you need to install Cloudera SQL Stream Builder in every namespace.

Examples

Note: Both examples require the namespaces to be created and available before running `helm install`

Installing Cloudera SQL Stream Builder in a single namespace (other than the Flink Operator)

If you want to have two namespaces, with

- Flink installed only in `namespace1` (but managing all namespaces)
- and Cloudera SQL Stream Builder installed only in `namespace2`,

you would use the following commands:

```
helm install \
--namespace namespace1 \
--set ssb.enabled=false
--set "flink-kubernetes-operator.watchNamespa
ces=namespace1,namespace2"
--set-file flink-kubernetes-operator.clouderaLicense.fileConte
nt=[***PATH TO LICENSE FILE***]
csa-operator csa-operator-1.0.0-b293.tgz
```

```
helm install \
--namespace namespace2 \
--set flink-kubernetes-operator.enabled=false
csa-operator csa-operator-1.0.0-b293.tgz
```

This example installs the Flink Operator in namespace1 without Cloudera SQL Stream Builder, and install Cloudera SQL Stream Builder (without the Flink Operator) in namespace2. The Flink Operator will be watching both namespaces, and the Helm chart is installing the Flink service account for both.

Installing Flink Operator and Cloudera SQL Stream Builder in separate namespaces

If you want to have two separate namespaces, with both Flink and Cloudera SQL Stream Builder installed and only watching a single namespace, you would use the following commands:

```
helm install \
--namespace namespace1 \
--set ssb.enabled=true
--set "flink-kubernetes-operator.watchNamespaces=namespace1"
--set-file flink-kubernetes-operator.clouderaLicense.fileContent=[***PATH TO LICENSE FILE***]
csa-operator csa-operator-1.0.0-b293.tgz
```

```
helm install \
--namespace namespace2 \
--set ssb.enabled=true
--set "flink-kubernetes-operator.watchNamespaces=namespace2"
--set-file flink-kubernetes-operator.clouderaLicense.fileContent=[***PATH TO LICENSE FILE***]
csa-operator csa-operator-1.0.0-b293.tgz
```

This example installs the Flink Operator, along with the service account, with Cloudera SQL Stream Builder in namespace1 and namespace1. The --set "flink-kubernetes-operator.watchNamespaces" parameter limits the Flink Operator's access to watch Flink Deployments in the single namespace it's installed in.

Updating Cloudera license

Cloudera Streaming Analytics - Kubernetes Operator requires a valid license to function. You must update expired licenses, otherwise, cluster resources will break down over time. Once the license expires, the cluster resources you deployed will continue to run, but reconciliation of resources will be blocked. For example: failed pods will not be restarted and deploying new Flink jobs will not be possible. In general, the control mechanisms in place that keep resources healthy will be blocked. This will result in deployed resources breaking down over time.

About this task

You register your initial license during installation by setting the clouderaLicense.fileContent Helm chart property. When this property is set, a Kubernetes secret is automatically generated that stores your license. The name of the secret is csa-op-license.

When the license expires, it must be updated. You can update the license by updating the secret that stores the license, with data from your new license, specifically the value of the data.license property in the secret.

Licenses can be updated at any time. If your license is already expired and you update your license, restrictions on functionality are lifted immediately after the license is updated.

Updating a license does not carry any risks and does not result in cluster downtime.

Before you begin



Important: Ensure that the start date of your new license is the current or a past date. Licenses become valid on their start date. Updating your old license with a new license that is not yet valid is the equivalent of registering an expired license. The start date of a license is specified in the startDate property of the license.

Procedure

1. Create a manifest in YAML format that defines the license secret.

Add your new license to `stringData.license`. Ensure that you add the full contents of the license as it is in the license file you received from Cloudera.

```
apiVersion: v1
kind: Secret
metadata:
  name: csa-op-license
type: Opaque
stringData:
  license: |
    [***YOUR LICENSE***]
```

2. Replace your old secret with the new one.

```
kubectl replace --namespace [***NAMESPACE***] -filename [***LICENSE SECRET
YAML***]
```

3. Verify that the license is updated.

```
kubectl get secret csa-op-license \
  --namespace [***NAMESPACE***] \
  --output jsonpath="{.data.license}" \
  | base64 --decode
```

The output of this command should be identical with the contents of the license file you received from Cloudera.