Cloudera Streams Messaging - Kubernetes Operator 1.4.0

Installation

Date published: 2024-06-11 Date modified: 2025-07-14



https://docs.cloudera.com/

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

Installation overview	4
Installing Strimzi with Helm	5
Installing Strimzi in an internet environment	6
Installing Strimzi in an air-gapped environment	
Installing Cloudera Surveyor for Apache Kafka with Helm	 11
Installing Cloudera Surveyor in an air-gapped environment	
Installing Cloudera Surveyor for evaluation	

Installation overview

Get started with installing Cloudera Streams Messaging - Kubernetes Operator. Learn about available installation methods, the installation artifacts, and where these artifacts are hosted.

Installing Cloudera Streams Messaging - Kubernetes Operator involves installing Strimzi and Cloudera Surveyor for Apache Kafka. Strimzi enables the deployment and management of Kafka and Kafka Connect clusters on Kubernetes, while Cloudera Surveyor is a UI application designed for monitoring and managing Kafka clusters.

Both Strimzi and Cloudera Surveyor can be installed using Helm. The two components are independent and if you are installing with Helm, you can choose to install only Strimzi or only Cloudera Surveyor. Whether you choose to install one or both components depends on your use case and operational objectives. Cloudera recommends installing both.

If you are on OpenShift, you can choose to install from OperatorHub as well. However, note that Cloudera Surveyor is not available for installation from OperatorHub. An installation from OperatorHub will only install Strimzi.

Installation artifacts and artifact locations

Cloudera Streams Messaging - Kubernetes Operator ships with various installation artifacts. These artifacts are hosted at two locations, the Cloudera Docker registry and the Cloudera Archive.

Both the Cloudera Docker registry and the Cloudera Archive require Cloudera credentials (username and password) for access. Credentials are provided to you as part of your license and subscription agreement. You can access both the registry and the archive using the same credentials.

Cloudera Docker registry – container.repository.cloudera.com

The Docker registry hosts the Helm chart as well as all Docker images used for installation. This includes Strimzi and Kafka, as well as Cloudera Surveyor artifacts.

Table 1: Strimzi and Kafka artifacts on the Cloudera Docker registry

Artifact	Location	Description
Strimzi Docker image	container.repository.cloudera.com/cl oudera/kafka-operator:0.45.0.1.4.0- b199	Docker image used for deploying Strimzi and its components.
Kafka Docker image	container.repository.cloudera.com/ cloudera/kafka:0.45.0.1.4.0-199- kafka-3.9.0.1.4	Docker image used for deploying Kafka and related components.
Strimzi Cluster Operator Helm chart	oci://container.repository.cloudera. com/cloudera-helm/csm-operator/s trimzi-kafka-operator:1.4.0-b199	Helm chart used to install the Strimzi Cluster Operator with helm install.

Table 2: Cloudera Surveyor artifacts on the Cloudera Docker registry

Artifact	Location	Description
Cloudera Surveyor server Docker image	container.repository.cloudera.com/cl oudera/surveyor:0.1.0.1.4.0-b199	Docker image used for deploying Cloudera Surveyor.
Cloudera Surveyor UI application Docker image	container.repository.cloudera.com/ cloudera/surveyor-app:0.1.0.1.4.0- b199	Docker image used for deploying Cloudera Surveyor.
Cloudera Surveyor Helm chart	oci://container.repository.cloudera. com/cloudera-helm/csm-operator/s urveyor:1.4.0-b199	Helm chart used to install Cloudera Surveyor with helm-install.

Cloudera Archive – archive.cloudera.com/p/csm-operator/

The Cloudera Archive hosts various installation artifacts including Helm charts, configuration examples, a YAML file containing all CRDs, diagnostic tools, and the maven artifacts.

Accessing the Cloudera Archive and the artifacts it hosts is not necessary to complete installation. All artifacts on the archive are supplemental resources. The following table collects the Cloudera Streams Messaging - Kubernetes Operator directories located in the archive with an overview of what artifacts they contain and how you can use them.

Table 3: Cloudera Streams Messaging - Kubernetes Operator directories on the Cloudera Archive

Archive Directory	Description
https://archive.cloudera.com/p/csm-operator/1.4/charts/	The charts directory contains the Helm charts for both Strimzi and Cloudera Surveyor. These are the same charts that are available on the Docker registry. Cloudera recommends that whenever possible you install with the charts hosted on the registry. The charts on the archive are provided in case you cannot access the registry or want to download the chart using a browser.
https://archive.cloudera.com/p/csm-operator/1.4/ examples/	The examples directory includes various examples of resource configuration files. You can use these to quickly deploy Kafka and other components in Kubernetes following installation.
https://archive.cloudera.com/p/csm-operator/1.4/install/	The install directory contains a single YAML file that collects all Strimzi Cluster Operator CRDs. The purpose of this file is twofold. One, the CRDs are rich in comments. Reviewing them can help you better understand how Kafka is deployed and managed with Strimzi in Cloudera Streams Messaging - Kubernetes Operator. It is a supplemental resource to the documentation. Two, this file is used to upgrade CRDs during upgrades. The CRDs are also included in the Strimzi Cluster Operator Helm chart, and Helm will automatically install the necessary CRDs to Kubernetes. You do not need to install them separately with the file hosted on the archive.
https://archive.cloudera.com/p/csm-operator/1.4/maven- repository/	The maven artifacts can be used to develop your own applications or tools for use with Cloudera Streams Messaging - Kubernetes Operator.
https://archive.cloudera.com/p/csm-operator/1.4/tools/	The tools directory contains command line tools that you use to collect diagnostic information and to troubleshoot cluster issues.

Installing Strimzi with Helm

Learn how to install Strimzi in Cloudera Streams Messaging - Kubernetes Operator with Helm. Installing Strimzi installs the applications and resources that enable you to deploy and manage Kafka in Kubernetes.

Strimzi is installed in your Kubernetes cluster with the Strimzi Cluster Operator Helm chart using the helm install command. When you install the chart, Helm installs the Strimzi Custom Resource Definitions (CRDs) included in Cloudera Streams Messaging - Kubernetes Operator and deploys the Strimzi Cluster Operator, which is an operator application that manages and monitors Kafka and related components. Additionally, other cluster resources and applications required for managing Kafka are also installed.

Installing Strimzi does not create or deploy a Kafka cluster. Kafka clusters are created following the installation by deploying Kafka and KafkaNodePool resources in the Kubernetes cluster with kubectl or oc.

Cloudera recommends that you install Strimzi once per Kubernetes cluster. Some resources are cluster-wide, which can cause issues if Strimzi is installed multiple times on the same cluster.

By default, the Strimzi Cluster Operator (deployed with installation) watches and manages the Kafka clusters that are deployed in the same namespace as the Strimzi Cluster Operator. However, you can configure it to watch any namespace. This allows you to manage multiple Kafka clusters deployed in different namespaces using a single installation.

Installation instructions are provided for the following scenarios.

- Installing in an internet environment Follow these steps to install Strimzi in a Kubernetes cluster with internet access.
- **Installing in an air-gapped environment** Follow these steps to install Strimzi in a Kubernetes cluster without internet access or if you want to install from a self-hosted registry.

Installing Strimzi in an internet environment

Complete these steps to install Strimzi if your Kubernetes cluster has internet access.

Before you begin

- Ensure that your Kubernetes environment meets requirements listed in System requirements.
- Your Kubernetes cluster requires internet connectivity to complete these steps. It must be able to reach the Cloudera Docker registry.
- 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 installation artifacts are hosted.
- Ensure that you have access to a valid Cloudera license.
- Review the Helm chart reference before installation.

The Helm chart accepts various configuration properties that you can set during installation. Using these properties you can customize your installation.

• If you are planning to watch and manage more than 20 Kafka clusters with a single installation, you must increase the memory and heap allocated to the Strimzi Cluster Operator. You can specify memory configuration in your helm install command. For more information, see Increasing Cluster Operator memory.

Procedure

1. Create a namespace in your Kubernetes cluster.

```
kubectl create namespace [***NAMESPACE***]
```

This is the namespace where you install Strimzi. Use the namespace you create in all installation steps that follow.

2. Prepare a Docker configuration JSON file that includes your Cloudera credentials.

```
{
  "auths": {
    "container.repository.cloudera.com": {
        "auth": "[***BASE64 ENCODED USERNAME:PASSWORD***]"
    }
}
```

The value of auth is the base64-encoded string of your credentials, with your username and password separated by a colon (:). For example, if your username is myuser and your password is pass, you must encode myuser:pass.

3. Create a Kubernetes Secret containing your Cloudera credentials.



Important: The Secret containing your credentials must exist in the namespace where you install Strimzi as well as all namespaces where you deploy Kafka or Kafka Connect clusters. Cloudera recommends that you create the Secret in all required namespaces now if you know what namespaces you will be using to deploy Kafka or Kafka Connect.

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

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

Enter your Cloudera credentials when prompted.

5. Install Strimzi with helm install.

```
helm install strimzi-cluster-operator \
    --namespace [***NAMESPACE***] \
    --set 'image.imagePullSecrets[0].name=[***CLOUDERA CREDENTIALS
    SECRET***]' \
    --set-file clouderaLicense.fileContent=[***PATH TO LICENSE FILE***] \
    --set watchAnyNamespace=true \
    oci://container.repository.cloudera.com/cloudera-helm/csm-operator/str
imzi-kafka-operator \
    --version 1.4.0-b199
```

- The string strimzi-cluster-operator is the Helm release name of the chart installation. This is an arbitrary, user defined name. Cloudera recommends that you use a unique and easily identifiable name.
- imagePullSecrets specifies what secret is used to pull images from the Cloudera registry. Setting this property is mandatory, otherwise, Helm cannot pull the necessary images from the Cloudera Docker registry. Ensure that you replace [***CLOUDERA CREDENTIALS SECRET***] with the name of the secret you created in Step 3 on page 7.
- clouderaLicense.fileContent is used to register your license. If this property is set, a secret is generated that contains the license you specify. Setting this property is mandatory. The Strimzi Cluster Operator will not function without a valid license. Ensure that you replace [***PATH TO LICENSE FILE***] with the full path to your Cloudera license file.
- You can use --set to set various other properties of the Helm chart. This enables you to customize your installation. For example, Cloudera recommends that you configure the Cluster Operator to watch all namespaces, this is configured by setting watchAnyNamespace to true. Alternatively, you can configure a list of specific namespaces to watch using watchNamespaces.
- 6. Verify your installation

This is done by listing the deployments and pods in your namespace. If installation is successful, you should see a strimzi-cluster-operator deployment and pod present in the cluster.

kubectl get deployments --namespace [***NAMESPACE***] UP-TO-DATE AVAILABLE AGE NAME READY #... strimzi-cluster-operator 1/11 1 13m kubectl get pods --namespace [***NAMESPACE***] NAME READY UP-TO-DATE AVAILABLE AGE

#... strimzi-cluster-operator 1/1 1 1 13m

7. Access supplemental resources available on the Cloudera Archive.

Supplemental resources available on the Cloudera Archive include various example files, diagnostic tools, and more. You can use these resources to quickly deploy Kafka clusters and to gain a better understanding of Strimzi and Cloudera Streams Messaging - Kubernetes Operator.

What to do next

- Deploy a Kafka cluster, see Deploying Kafka.
- Set up Prometheus for monitoring, see Configuring Kafka for Prometheus monitoring and Monitoring with Prometheus.

Related Information

Cloudera Archive

Installing Strimzi in an air-gapped environment

Complete these steps to install Strimzi if your Kubernetes cluster does not have internet access or if you want to install from a self-hosted registry.

Before you begin

- Ensure that your Kubernetes environment meets requirements listed in System requirements.
- A self-hosted Docker registry is required. Your registry must be accessible by your Kubernetes cluster.
- A machine with Internet connectivity is required. While the Kubernetes cluster does not need internet access, you will need a machine to pull the images from the Cloudera Docker registry.
- Access to docker or equivalent utility that you can use to pull and push images is required. The following steps use docker. Replace commands where necessary.
- 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 installation artifacts are hosted.
- Ensure that you have access to a valid Cloudera license.
- Review the Helm chart reference before installation.

The Helm chart accepts various configuration properties that you can set during installation. Using these properties you can customize your installation.

• If you are planning to watch and manage more than 20 Kafka clusters with a single installation, you must increase the memory and heap allocated to the Strimzi Cluster Operator. You can specify memory configuration in your helm install command. For more information, see Increasing Cluster Operator memory.

Procedure

1. Copy the following installation artifacts to your self-hosted registry.

Table 4: Strimzi and Kafka artifacts on the Cloudera Docker registry

Artifact	Location	Description
Strimzi Docker image	container.repository.cloudera.com/cloudera/k afka-operator:0.45.0.1.4.0-b199	Docker image used for deploying Strimzi and its components.
Kafka Docker image	container.repository.cloudera.com/cloudera/k afka:0.45.0.1.4.0-199-kafka-3.9.0.1.4	Docker image used for deploying Kafka and related components.

Artifact	Location	Description
Strimzi Cluster Operator Helm chart	oci://container.repository.cloudera.com/clou dera-helm/csm-operator/strimzi-kafka-ope rator:1.4.0-b199	Helm chart used to install the Strimzi Cluster Operator with helm install.

This step involves pulling the artifacts from the Cloudera Docker registry, retagging them, and then pushing them to your self-hosted registry. The exact steps you need to carry it out depend on your environment and how your registry is set up. The following substeps demonstrate the basic workflow using docker and helm.

a) Log in to the Cloudera Docker registry with both docker and helm.

Provide your Cloudera credentials when prompted.

docker login container.repository.cloudera.com

helm registry login container.repository.cloudera.com

b) Pull the Docker images from the Cloudera Docker registry.

```
docker pull \
    container.repository.cloudera.com/cloudera/[***IMAGE
    NAME***]:[***VERSION***]
```

c) Pull the Strimzi Cluster Operator Helm chart.

```
helm pull \
    oci://container.repository.cloudera.com/cloudera-helm/csm-operator/str
imzi-kafka-operator \
    --version 1.4.0-b199
```

d) Retag the Docker images you pulled so that they contain the address of your registry.

```
docker tag \
  [***ORIGINAL IMAGE TAG***] \
  [***YOUR REGISTRY***]/cloudera/[***IMAGE NAME***]:[***VERSION***]
```

e) Push the images and chart to your self-hosted registry.

```
docker push \
  [***YOUR REGISTRY***]/cloudera/[***IMAGE NAME***]:[***VERSION***]
```

```
helm push \
strimzi-kafka-operator-1.4.0-b199.tgz \
oci://[***REGISTRY HOSTNAME***]:[***PORT***]/cloudera-helm/csm-opera
tor/
```

2. Create a namespace in your Kubernetes cluster.

```
kubectl create namespace [***NAMESPACE***]
```

This is the namespace where you install Strimzi. Use the namespace you create in all installation steps that follow. **3.** Prepare a Docker configuration JSON file that includes credentials for your self-hosted registry.

```
{
    "auths": {
        "[****YOUR REGISTRY***]": {
            "auth": "[***BASE64 ENCODED USERNAME:PASSWORD***]"
        }
}
```

}

The value of auth is the base64-encoded string of your credentials, with your username and password separated by a colon (:). For example, if your username is myuser and your password is pass, you must encode myuser:pass.

4. Create a Kubernetes Secret containing credentials for your self-hosted registry.



Important: The Secret containing your credentials must exist in the namespace where you install Strimzi as well as all namespaces where you deploy Kafka or Kafka Connect clusters. Cloudera recommends that you create the Secret in all required namespaces now if you know what namespaces you will be using to deploy Kafka or Kafka Connect.

5. Log in to your self-hosted registry with helm.

```
helm registry login [***REGISTRY HOSTNAME***]:[***PORT***]
```

Enter your credentials when prompted.

6. Install Strimzi with helm install.

```
helm install strimzi-cluster-operator \
    --namespace [***NAMESPACE***] \
    --set 'image.imagePullSecrets[0].name=[***REGISTRY CREDENTIALS
    SECRET***]' \
    --set defaultImageRegistry=[***REGISTRY HOSTNAME***]:[***PORT***] \
    --set-file clouderaLicense.fileContent=[***PATH TO LICENSE FILE***] \
    oci://[***YOUR REGISTRY***]/cloudera-helm/csm-operator/strimzi-kafka-op
erator \
    --version 1.4.0-b199 \
    --set watchAnyNamespace=true
```

- The string strimzi-cluster-operator is the Helm release name of the chart installation. This is an arbitrary, user defined name. Cloudera recommends that you use a unique and easily identifiable name.
- imagePullSecrets specifies what secret is used to pull images from the specified registry. Ensure that you replace [***REGISTRY CREDENTIALS SECRET***] with the name of the secret you created in Step 4 on page 10.
- clouderaLicense.fileContent is used to register your license. If this property is set, a secret is generated that contains the license you specify. Setting this property is mandatory. The Strimzi Cluster Operator will not function without a valid license. Ensure that you replace [***PATH TO LICENSE FILE***] with the full path to your Cloudera license file.
- You can use --set to set various other properties of the Helm chart. This enables you to customize your installation. For example, Cloudera recommends that you configure the Cluster Operator to watch all namespaces, this is configured by setting watchAnyNamespace to true. Alternatively, you can configure a list of specific namespaces to watch using watchNamespaces.
- 7. Verify your installation

This is done by listing the deployments and pods in your namespace. If installation is successful, you should see a strimzi-cluster-operator deployment and pod present in the cluster.

```
kubectl get deployments --namespace [***NAMESPACE***]
NAME READY UP-TO-DATE AVAILABLE AGE
#...
```

strimzi-cluster-operator	1/1	1	1	13m
kubectl get podsnamespa	ace [***]	IAMESPACE***]		
NAME #	READY	UP-TO-DATE	AVAILABLE	AGE
strimzi-cluster-operator	1/1	1	1	13m

8. Access supplemental resources available on the Cloudera Archive.

Supplemental resources available on the Cloudera Archive include various example files, diagnostic tools, and more. You can use these resources to quickly deploy Kafka clusters and to gain a better understanding of Strimzi and Cloudera Streams Messaging - Kubernetes Operator.

What to do next

- Deploy a Kafka cluster, see Deploying Kafka.
- Set up Prometheus for monitoring, see Configuring Kafka for Prometheus monitoring and Monitoring with Prometheus.

Related Information

Cloudera Archive

Installing Cloudera Surveyor for Apache Kafka with Helm

Learn how to install Cloudera Surveyor in Cloudera Streams Messaging - Kubernetes Operator with Helm. Cloudera Surveyor is a UI application that enables you to manage and monitor Kafka clusters.

Cloudera Surveyor Is installed in your Kubernetes cluster with the Cloudera Surveyor Helm chart using the helm install command. When you install the chart, Helm deploys an instance of Cloudera Surveyor, which enables you to manage and monitor your Kafka clusters through a UI interface.

During installation, you configure Cloudera Surveyor using a custom values file (values.yaml) passed to the Helm chart with the --values (-f) option. This file contains properties for configuring Cloudera Surveyor itself as well as Kafka cluster-specific settings that define which Kafka clusters Cloudera Surveyor connects to. Additionally, some properties are configured with --set options.

Installation instructions are provided for the following scenarios.

- **Installing in an internet environment** Follow these steps to install a fully secure instance of Cloudera Surveyor in a Kubernetes cluster with internet access.
- **Installing in an air-gapped environment** –Follow these steps to install a fully secure instance of Cloudera Surveyor in a Kubernetes cluster without internet access or if you want to install from a self-hosted registry.
- **Installing for evaluation** Follow these steps to install an unsecure instance of Cloudera Surveyor for development or proof of concept purposes.

Installing Cloudera Surveyor in an internet environment

Complete these steps to install Cloudera Surveyor if your Kubernetes cluster has internet access. These steps install a fully secure instance of Cloudera Surveyor that has authentication, authorization, and channel encryption configured. The example configuration also demonstrates how you can connect a secure Kafka cluster to Cloudera Surveyor.

Before you begin

- General prerequisites:
 - Ensure that your Kubernetes environment meets requirements listed in System requirements.
 - Your Kubernetes cluster requires internet connectivity to complete these steps. It must be able to reach the Cloudera Docker registry.
 - 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 installation artifacts are hosted.
 - Ensure that you have access to a valid Cloudera license.
 - Review the Helm chart reference before installation.

The Helm chart accepts various configuration properties that you can set during installation. Using these properties you can customize your installation.

- Prerequisites for channel encryption (TLS):
 - An Ingress controller is installed in your Kubernetes cluster. These steps use the Ingress-Nginx controller.
 - Optional: cert-manager is installed in your Kubernetes cluster.
- Prerequisites for LDAP authentication:
 - An LDAP server is available that has TLS enabled.
 - The server is accessible from the Kubernetes cluster where Cloudera Surveyor is deployed.
 - Entries containing usernames and passwords are located under a common base in the directory information tree. Passwords must be stored in the userPassword attribute in the user entries.
- Prerequisites for authorization:
 - Ensure that Kafka ACLs are set up for your Kafka cluster. Cloudera Surveyor uses Kafka ACLs to provide authorization.

Procedure

1. Create a namespace in your Kubernetes cluster.

```
kubectl create namespace [***NAMESPACE***]
```

This is the namespace where you install Cloudera Surveyor. Use the namespace you create in all installation steps that follow.

2. Prepare a Docker configuration JSON file that includes your Cloudera credentials.

```
{
   "auths": {
    "container.repository.cloudera.com": {
        "auth": "[***BASE64 ENCODED USERNAME:PASSWORD***]"
     }
}
```

The value of auth is the base64-encoded string of your credentials, with your username and password separated by a colon (:). For example, if your username is myuser and your password is pass, you must encode myuser:pass.

3. Create a Kubernetes Secret containing your Cloudera credentials.

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

helm registry login container.repository.cloudera.com

Enter your Cloudera credentials when prompted.

5. Create Secrets for sensitive Kafka client configuration values.

Cloudera Surveyor connects to Kafka clusters as any other Kafka client and requires a client configuration. If the Kafka cluster is secured, the client configuration will include sensitive property values. Cloudera recommends that you store sensitive values in Secrets, mount the Secrets to the Cloudera Surveyor Container, and reference the values in your configuration instead of hard-coding them.

Typically, you need to create two Secrets. One contains the Kafka cluster truststore and password, and the other contains a JAAS configuration.

```
kubectl create secret generic [***KAFKA TRUSTSTORE SECRET***] \
    --namespace [***NAMESPACE***] \
    --from-file=[***KAFKA TRUSTSTORE KEY***]=[***PATH TO TRUSTSTORE***] \
    --from-file=[***KAFKA TRUSTSTORE PASSWORD KEY***]=[***TRUSTSTORE
    PASSWORD FILE***]
```

```
kubectl create secret generic [***KAFKA JAAS.CONF SECRET***] \
    --namesapce [***NAMESPACE***] \
    --from-file=[***KAFKA JAAS.CONF KEY***]=[***PATH TO JAAS.CONF***]
```

Take note of the Secret names as well as the key names you configure. You will need to specify them in a later step.

- 6. Prepare a certificate and private key for Cloudera Surveyor.
 - If you have cert-manager available, create a Certificate resource. Take note of the Secret name you configure in spec.secretname of the Certificate resource, you will need to specify it in a later step.
 - If you are managing keys manually, create a certificate and private key and save it to a Secret. Take note of the Secret name, you will need to specify it in a later step.

This Secret is referred to as [***SURVEYOR TLS CERT SECRET ***] in the following steps.

- 7. Prepare a certificate and private key for Ingress.
 - If you have cert-manager available, the certificate and private key for Ingress are automatically requested by the Ingress. You only need to ensure that you have a valid Issuer available in cert-manager. You specify the name of the Issuer resource in a later step.
 - If you are managing keys manually, create a certificate and private key and save it to a Secret. Take note of the Secret name, you will need to specify it in a later step.

This Secret referred to as [***INGRESS TLS CERT SECRET***] in the following steps.

- **8.** Set up resources for LDAP authentication.
 - a) Generate an authentication key and save it as a file.

The authentication key is a random sequence of at least 32 bytes that is saved to a file. To generate it, use any tool or method that is available to you. Take note of the location and name of the file that you create. You will need to specify the file in a later step.

b) Generate a Java truststore (PKCS12 or JKS) containing the TLS certificate of the root Certificate Authority (CA) of the LDAP certificate chain.

```
keytool -import -trustcacerts -file [***LDAP ROOT CA***] \
   -keystore [***TRUSTSTORE NAME***] \
   -storepass [***TRUSTSTORE PASSWORD***] \
   -storetype PKCS12
```

c) Create a Secret containing the truststore and the truststore password.

```
kubectl create secret generic [***LDAP TRUSTSTORE SECRET***] \
    --namespace [***NAMESPACE***] \
    --from-file=[***TRUSTSTORE SECRET KEY***]=[***TRUSTSTORE NAME***] \
    --from-file=[***TRUSTSTORE PW SECRET KEY***]=[***PATH TO TRUSTSTORE PW
FILE***]
```

Take note of [***LDAP TRUSTSTORE SECRET***], [***TRUSTSTORE SECRET KEY***], and [***TRUSTSTORE PW SECRET KEY***]. You will need to specify these names in a custom values file you create in a later step.

9. Prepare a custom values file (values.yaml).

The values file contains configuration for Cloudera Surveyor. This file specifies the Kafka clusters that Cloudera Surveyor connects to as well as various other configuration properties.

```
clusterConfigs:
  clusters:
    - clusterName: [***CLUSTER NAME***]
      tags:
        - [***TAG1***]
        - [***TAG2***]
      bootstrapServers: [***BOOTSTRAP SERVERS***]
      commonClientConfig:
        security.protocol: "SASL_SSL"
        sasl.mechanism: PLAIN
        ssl.truststore.type: "pkcs12"
        ssl.truststore.location: "/opt/secrets/[***KAFKA TRUSTSTORE
SECRET***]/[***KAFKA TRUSTSTORE FILE***]"
        ssl.truststore.password: "\\${dir:/opt/secrets/[***KAFKA
TRUSTSTORE SECRET***]:[***KAFKA TRUSTSTORE PASSWORD FILE***]}"
       sasl.jaas.config: "\\${dir:/opt/secrets/[***KAFKA JAAS.CONF
 SECRET***]:[***KAFKA JAAS.CONF***]}"
      adminOperationTimeout: PT1M
      authorization:
        enabled: true
secretsToMount:
  - create: false
   secretRef: [***KAFKA TRUSTSTORE SECRET***]
   items:
      - key: [***KAFKA TRUSTSTORE KEY***]
        path: [***KAFKA TRUSTSTORE FILE***]
      - key: [***KAFKA TRUSTSTORE PASSWORD KEY***]
        path: [***KAFKA TRUSTSTORE PASSWORD FILE***]
  - create: false
   secretRef: [***KAFKA JAAS.CONF SECRET***]
   items:
      - key: [***KAFKA JAAS.CONF KEY***]
        path: [***KAFKA JAAS.CONF***]
surveyorConfig:
  surveyor:
   authentication:
      enabled: true
  quarkus:
   security:
      ldap:
        dir-context:
          url: ldaps://openldap-chart.ldap:1390
          principal: cn=admin,dc=openldap-chart,dc=ldap
          password: adminpassword
        identity-mapping:
          rdn-identifier: uid
```

```
search-base-dn: ou=users,dc=openldap-chart,dc=ldap
          attribute-mappings:
            "0":
              from: cn
              filter: (cn={0},ou=users,dc=openldap-chart,dc=ldap)
              filter-base-dn: ou=users,dc=openldap-chart,dc=ldap
tlsConfigs:
  enabled: true
  secretRef: [***SURVEYOR TLS CERT SECRET***]
ingress:
 enabled: true
 protocol: HTTPS
 className: nginx
 rules:
   path: "/"
   host: "MY-DOMAIN.EXAMPLE.COM"
   port: 8443
  tls:
    enabled: true
    secretRef: "[***INGRESS TLS CERT SECRET***]"
globalTruststore:
  secretRef:
   name: [***LDAP TRUSTSTORE SECRET***]
   key: [***TRUSTSTORE SECRET KEY***]
  type: PKCS12
 password:
   secretRef:
      name: [***LDAP TRUSTSTORE SECRET***]
      key: [***TRUSTSTORE PW SECRET KEY***]
```

For clusterConfigs

clusterConfigs specifies the Kafka clusters that Cloudera Surveyor connects to. Clusters specified here are the ones that will be available on the UI for monitoring and management.

- clusterConfigs.clusters[*] An array of Kafka clusters and their configuration. Each entry defines the configuration for a single Kafka cluster.
- clusterConfigs.clusters[*].clustername The name of the cluster. This name is displayed on the UI.
- clusterConfigs.clusters[*].bootstrapServers A comma-separated list of the bootstrap servers for the Kafka cluster that Cloudera Surveyor connects to. Specify multiple servers for highly available connections.
- clusterConfigs.clusters[*].tags User defined tags. Used for organization and filtering.
- clusterConfigs.clusters[*].commonClientConfig Kafka client configuration properties applied to all clients for this cluster. Must contain upstream Kafka client properties as a map. The exact properties that you specify here depend on the security configuration of the Kafka cluster that you want to connect. This example specifies a Kafka cluster that uses PLAIN authentication with TLS.

Sensitive property values are referenced from Secrets instead of being hardcoded. Secrets containing sensitive properties are mounted using the secretsToMount property. References use Kafka DirectoryConfigProvider syntax. For more information, see *Registering Kafka clusters*.

• clusterConfigs.clusters[*].authorization.enabled – Enables authorization for this cluster.

For secretsToMount

secretsToMount specifies the Secrets to mount in the Cloudera Surveyor Container. You use secretsT oMount to mount the sensitive values required for Kafka client configuration specified in clusterConfigs.

- secretsToMount[*].create Specifies whether to create the Secret. Set to false in this example as the Secrets are assumed to already exist.
- secretsToMount[*].secretRef The name of the Secret to mount.
- secretsToMount[*].items[*].key The key in the Secret to mount.

 secretsToMount[*].items[*].path – The path where the item is mounted. The path is relative to /opt/ secrets/[***SECRET NAME***]/ in the Cloudera Surveyor Container.

For surveyorConfig

surveyorConfig specifies global configuration for Cloudera Surveyor. This example sets various authentication properties.

- surveyorConfig.surveyor.authentication.enabled Enables or disables authentication. Set to true by default. Included in the example as a reference, you do not need to set the property explicitly to enable authentication.
- surveyorconfig.surveyor.authentication.* These properties control user session behavior and authentication token management. They define the session timeout, inactivity timeout, and the interval for renewing authentication tokens. All time intervals are specified in ISO 8601 format. For optimal security, Cloudera recommends that token renewal is shorter than or equal to half of the inactivity timeout.
- surveyorconfig.quarkus.security.ldap.dircontext.* These properties configure the LDAP server that Cloudera Surveyor connects to. They specify the server URL, the distinguished name (DN) of the bind user, and the password of the bind user. These are required for establishing a secure connection with the LDAP directory.



Important: To ensure that Cloudera Surveyor connects to the LDAP server securely, the URL you specify in surveyorconfig.quarkus.security.ldap.dir-context.url must start with ldaps://.

• surveyorconfig.quarkus.security.ldap.identity-mapping.* - These properties configure how Cloudera Surveyor interacts with the LDAP directory to identify users and map their group memberships. They define the attributes and base DNs used to locate user entries and groups in the directory, as well as the filters applied to verify group membership.



Note: Groups are not required for authentication. You might want to configure the surveyorConf ig.quarkus.security.ldap.identity-mapping.attribute-mappings.* properties accordingly. For more information on the surveyorConfig.quarkus.* properties, see *Using Security with an LDAP Realm* in the Quarkus documentation.

For tlsConfigs

tlsConfigs enables TLS and specifies the Secret containing the certificate of Cloudera Surveyor.

- tlsConfigs.enabled Enables or disables TLS. Set to true by default. Included in the example as a reference, you do not need to set the property explicitly to enable TLS.
- tlsConfigs.secretRef Name of the Secret containing the Cloudera Surveyor certificate and key.

For ingress

ingress enables the creation of an Ingress. The Ingress provides secure external access to the Cloudera Surveyor UI.

- ingress.enabled Enables or disables Ingress.
- ingress.protocol Configures the Ingress protocol.
- ingress.className The class name of the Ingress controller. This example configures the Ingress-Nginx controller.
- ingress.rules.host Specifies the DNS hostname that the Ingress controller should match for incoming HTTP/HTTPS requests.
- ingress.rules.port The port of the Ingress rule. This is the port of the Kubernetes Service that the Ingress forwards requests to.
- ingress.tls.enabled Enables TLS for the Ingress.

• ingress.tls.secretRef – The name of the Secret that contains Ingress TLS certificates.

Note: If you are using cert-manager, configure ingress.tls.issuer and set it to the name of your Issuer resource. If set, Ingress will automatically request a certificate and save it to the Secret specified in ingress.tls.secretRef.

For globalTruststore

globalTruststore specifies the Secrets containing the truststore of the LDAP server and the password for the truststore.

- globalTruststore.secretRef.name The name of the Kubernetes Secret containing the truststore of the LDAP server.
- globalTruststore.secretRef.key The key in the Kubernetes Secret that contains the truststore.
- globalTruststore.password.name The name of the Kubernetes Secret containing the truststore password.
- globalTruststore.password.key The key in the Kubernetes Secret that contains the truststore password.

10. Install Cloudera Surveyor with helm install.

```
helm install cloudera-surveyor \
    --namespace [***NAMESPACE***] \
    --values [***VALUES FILE***] \
    --set-file surveyorConfig.surveyor.authentication.keys.active=[***PATH
    TO AUTHENTICATION KEY FILE***] \
    --set 'image.imagePullSecrets=[***CLOUDERA CREDENTIALS SECRET***] '
    --set-file clouderaLicense.fileContent=[***PATH TO LICENSE FILE***] \
    oci://container.repository.cloudera.com/cloudera-helm/csm-operator/surv
eyor \
    --version 1.4.0-b199
```

- The string cloudera-surveyor is the Helm release name of the chart installation. This is an arbitrary, user defined name. Cloudera recommends that you use a unique and easily identifiable name.
- [***VALUES FILE***] is the values file you prepared in Step 9 on page 14.
- surveyorConfig.surveyor.authentication.keys.active specifies the file containing the authentication key that you generated in Step 8 on page 13. The key is required for LDAP authentication to function.
- imagePullSecrets specifies what Secret is used to pull images from the Cloudera registry. Setting this property is mandatory, otherwise, Helm cannot pull the necessary images from the Cloudera Docker registry. Ensure that you replace [***CLOUDERA CREDENTIALS SECRET***] with the name of the Secret you created in Step 3 on page 12.
- clouderaLicense.fileContent is used to register your license. If this property is set, a Secret is generated that contains the license you specify. Setting this property is mandatory. Cloudera Surveyor will not function without a valid license. Ensure that you replace [***PATH TO LICENSE FILE***] with the full path to your Cloudera license file.
- You can use --set to override properties that are defined in your values file, or add additional properties that are not present in your values file.

11. Verify your installation.

This is done by listing the Deployments and Pods in your namespace. If installation is successful, a Cloudera Surveyor Deployment and two Pods will be present in the cluster.

```
kubectl get deployments --namespace [***NAMESPACE***]
NAME READY UP-TO-DATE AVAILABLE AGE
#...
```

cloudera-surveyor	2/2	2	2		13m		
kubectl get podsr	lamespac	e [***N.	AMESPACE*	**]			
NAME		READY	STATUS	RESTARTS	AGE		
#							
cloudera-surveyor-64	19f755f6	d-689ax	1/1	Running		0	
12m	19175510	a oojgn	±/±	Raiming		0	
1.5111						_	
cloudera-surveyor-64	19±755±6	d-xj2kp	1/1	Running		0	
1 3 m							

12. Access the Cloudera Surveyor UI.

The UI is accessible by connecting to the Ingress.

```
kubectl get ingress cloudera-surveyor-ingress --namespac
e [***NAMESPACE***]
```

NAME	CLASS	HOSTS	ADDRES	S
cloudera-surveyor-ingress 443	nginx	my-domain.example.com	10.14.91.1	80,

Typically you will be able to access the UI through the host and port listed. However, the exact port that you have to use might be infrastructure dependent. If you are unable to connect, check the configuration of your Ingress controller. You can also try connecting directly to the Service of the Ingress.

Results

Cloudera Surveyor is installed. You can now manage and monitor your Kafka clusters using the UI. Related Information Registering Kafka clusters

Using Security with an LDAP Realm | Quarkus

Installing Cloudera Surveyor in an air-gapped environment

Complete these steps to install Cloudera Surveyor if your Kubernetes cluster does not have internet access or if you want to install from a self-hosted registry. These steps install a fully secure instance of Cloudera Surveyor that has authentication, authorization, and channel encryption configured. The example configuration also demonstrates how you can connect a secure Kafka cluster to Cloudera Surveyor.

Before you begin

- General prerequisites:
 - Ensure that your Kubernetes environment meets requirements listed in System requirements.
 - A self-hosted Docker registry is required. Your registry must be accessible by your Kubernetes cluster.
 - A machine with Internet connectivity is required. While the Kubernetes cluster does not need internet access, you will need a machine to pull the images from the Cloudera Docker registry.
 - Access to docker or equivalent utility that you can use to pull and push images is required. The following steps use docker. Replace commands where necessary.
 - 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 installation artifacts are hosted.
 - Ensure that you have access to a valid Cloudera license.
 - Review the Helm chart reference before installation.

The Helm chart accepts various configuration properties that you can set during installation. Using these properties you can customize your installation.

- Prerequisites for channel encryption (TLS):
 - An Ingress controller is installed in your Kubernetes cluster. These steps use the Ingress-Nginx controller.
 - Optional: cert-manager is installed in your Kubernetes cluster.
- Prerequisites for LDAP authentication:
 - An LDAP server is available that has TLS enabled.
 - The server is accessible from the Kubernetes cluster where Cloudera Surveyor is deployed.
 - Entries containing usernames and passwords are located under a common base in the directory information tree. Passwords must be stored in the userPassword attribute in the user entries.
- Prerequisites for authorization:
 - Ensure that Kafka ACLs are set up for your Kafka cluster. Cloudera Surveyor uses Kafka ACLs to provide authorization.

Procedure

1. Copy the following installation artifacts to your self-hosted registry.

Table 5: Cloudera Surveyor artifacts on the Cloudera Docker registry

Artifact	Location	Description
Cloudera Surveyor server Docker image	container.repository.cloudera.com/cloudera/s urveyor:0.1.0.1.4.0-b199	Docker image used for deploying Cloudera Surveyor.
Cloudera Surveyor UI application Docker image	container.repository.cloudera.com/cloudera/s urveyor-app:0.1.0.1.4.0-b199	Docker image used for deploying Cloudera Surveyor.

Artifact	Location	Description
Cloudera Surveyor Helm chart	oci://container.repository.cloudera.com/clou dera-helm/csm-operator/surveyor:1.4.0-b199	Helm chart used to install Cloudera Surveyor with helm-install.

This step involves pulling the artifacts from the Cloudera Docker registry, retagging them, and then pushing them to your self-hosted registry. The exact steps you need to carry it out depend on your environment and how your registry is set up. The following substeps demonstrate the basic workflow using docker and helm.

a) Log in to the Cloudera Docker registry with both docker and helm.

Provide your Cloudera credentials when prompted.

```
docker login container.repository.cloudera.com
```

helm registry login container.repository.cloudera.com

b) Pull the Docker images from the Cloudera Docker registry.

```
docker pull \
    container.repository.cloudera.com/cloudera/[***IMAGE
    NAME***]:[***VERSION***]
```

c) Pull the Cloudera Surveyor Helm chart.

```
helm pull \
    oci://container.repository.cloudera.com/cloudera-helm/csm-operator/sur
veyor \
    --version 1.4.0-b199
```

d) Retag the Docker images you pulled so that they contain the address of your registry.

```
docker tag \
  [***ORIGINAL IMAGE TAG***] \
  [***YOUR REGISTRY***]/cloudera/[***IMAGE NAME***]:[***VERSION***]
```

e) Push the images and chart to your self-hosted registry.

```
docker push \
  [***YOUR REGISTRY***]/cloudera/[***IMAGE NAME***]:[***VERSION***]
```

```
helm push \
   surveyor-1.4.0-b199.tgz \
   oci://[***REGISTRY HOSTNAME***]:[***PORT***]/cloudera-helm/csm-opera
tor/surveyor
```

2. Create a namespace in your Kubernetes cluster.

kubectl create namespace [***NAMESPACE***]

This is the namespace where you install Cloudera Surveyor. Use the namespace you create in all installation steps that follow.

3. Prepare a Docker configuration JSON file that includes credentials for your self-hosted registry.

```
{
    "auths": {
        "[****YOUR REGISTRY***]": {
            "auth": "[***BASE64 ENCODED USERNAME:PASSWORD***]"
        }
}
```

}

The value of auth is the base64-encoded string of your credentials, with your username and password separated by a colon (:). For example, if your username is myuser and your password is pass, you must encode myuser:pass.

4. Create a Kubernetes Secret containing credentials for your self-hosted registry.

5. Log in to your self-hosted registry with helm.

helm registry login [***REGISTRY HOSTNAME***]:[***PORT***]

Enter your credentials when prompted.

6. Create Secrets for sensitive Kafka client configuration values.

Cloudera Surveyor connects to Kafka clusters as any other Kafka client and requires a client configuration. If the Kafka cluster is secured, the client configuration will include sensitive property values. Cloudera recommends that you store sensitive values in Secrets, mount the Secrets to the Cloudera Surveyor Container, and reference the values in your configuration instead of hard-coding them.

Typically, you need to create two Secrets. One contains the Kafka cluster truststore and password, and the other contains a JAAS configuration.

```
kubectl create secret generic [***KAFKA TRUSTSTORE SECRET***] \
    --namespace [***NAMESPACE***] \
    --from-file=[***KAFKA TRUSTSTORE KEY***]=[***PATH TO TRUSTSTORE***] \
    --from-file=[***KAFKA TRUSTSTORE PASSWORD KEY***]=[***TRUSTSTORE
    PASSWORD FILE***]
```

```
kubectl create secret generic [***KAFKA JAAS.CONF SECRET***] \
    --namesapce [***NAMESPACE***] \
    --from-file=[***KAFKA JAAS.CONF KEY***]=[***PATH TO JAAS.CONF***]
```

Take note of the Secret names as well as the key names you configure. You will need to specify them in a later step.

- 7. Prepare a certificate and private key for Cloudera Surveyor.
 - If you have cert-manager available, create a Certificate resource. Take note of the Secret name you configure in spec.secretname of the Certificate resource, you will need to specify it in a later step.
 - If you are managing keys manually, create a certificate and private key and save it to a Secret. Take note of the Secret name, you will need to specify it in a later step.

This Secret is referred to as [***SURVEYOR TLS CERT SECRET***] in the following steps.

- 8. Prepare a certificate and private key for Ingress.
 - If you have cert-manager available, the certificate and private key for Ingress are automatically requested by the Ingress. You only need to ensure that you have a valid Issuer available in cert-manager. You specify the name of the Issuer resource in a later step.
 - If you are managing keys manually, create a certificate and private key and save it to a Secret. Take note of the Secret name, you will need to specify it in a later step.

This Secret referred to as [***INGRESS TLS CERT SECRET***] in the following steps.

- 9. Set up resources for LDAP authentication.
 - a) Generate an authentication key and save it as a file.

The authentication key is a random sequence of at least 32 bytes that is saved to a file. To generate it, use any tool or method that is available to you. Take note of the location and name of the file that you create. You will need to specify the file in a later step.

b) Generate a Java truststore (PKCS12 or JKS) containing the TLS certificate of the root Certificate Authority (CA) of the LDAP certificate chain.

```
keytool -import -trustcacerts -file [***LDAP ROOT CA***] \
   -keystore [***TRUSTSTORE NAME***] \
   -storepass [***TRUSTSTORE PASSWORD***] \
   -storetype PKCS12
```

c) Create a Secret containing the truststore and the truststore password.

```
kubectl create secret generic [***LDAP TRUSTSTORE SECRET***] \
    --namespace [***NAMESPACE***] \
    --from-file=[***TRUSTSTORE SECRET KEY***]=[***TRUSTSTORE NAME***] \
    --from-file=[***TRUSTSTORE PW SECRET KEY***]=[***PATH TO TRUSTSTORE PW
FILE***]
```

```
Take note of [***LDAP TRUSTSTORE SECRET***], [***TRUSTSTORE SECRET KEY***], and [***TRUSTSTORE PW SECRET KEY***]. You will need to specify these names in a custom values file you create in a later step.
```

10. Prepare a custom values file (values.yaml).

The values file contains configuration for Cloudera Surveyor. This file specifies the Kafka clusters that Cloudera Surveyor connects to as well as various other configuration properties.

```
clusterConfigs:
  clusters:
    - clusterName: [***CLUSTER NAME***]
      taqs:
        - [***TAG1***]
        - [***TAG2***]
      bootstrapServers: [***BOOTSTRAP SERVERS***]
      commonClientConfig:
        security.protocol: "SASL_SSL"
        sasl.mechanism: PLAIN
        ssl.truststore.type: "pkcs12"
        ssl.truststore.location: "/opt/secrets/[***KAFKA TRUSTSTORE
 SECRET***]/[***KAFKA TRUSTSTORE FILE***]"
        ssl.truststore.password: "\\${dir:/opt/secrets/[***KAFKA
TRUSTSTORE SECRET***]:[***KAFKA TRUSTSTORE PASSWORD FILE***]}"
        sasl.jaas.config: "\\${dir:/opt/secrets/[***KAFKA JAAS.CONF
 SECRET***]:[***KAFKA JAAS.CONF***]}"
      adminOperationTimeout: PT1M
      authorization:
        enabled: true
secretsToMount:
  - create: false
   secretRef: [***KAFKA TRUSTSTORE SECRET***]
   items:
      - key: [***KAFKA TRUSTSTORE KEY***]
       path: [***KAFKA TRUSTSTORE FILE***]
      - key: [***KAFKA TRUSTSTORE PASSWORD KEY***]
        path: [***KAFKA TRUSTSTORE PASSWORD FILE***]
  - create: false
   secretRef: [***KAFKA JAAS.CONF SECRET***]
    items:
      - key: [***KAFKA JAAS.CONF KEY***]
```

```
path: [***KAFKA JAAS.CONF***]
surveyorConfig:
  surveyor:
    authentication:
      enabled: true
  quarkus:
    security:
      ldap:
        dir-context:
          url: ldaps://openldap-chart.ldap:1390
          principal: cn=admin,dc=openldap-chart,dc=ldap
          password: adminpassword
        identity-mapping:
          rdn-identifier: uid
          search-base-dn: ou=users,dc=openldap-chart,dc=ldap
          attribute-mappings:
            "0":
              from: cn
              filter: (cn={0},ou=users,dc=openldap-chart,dc=ldap)
              filter-base-dn: ou=users,dc=openldap-chart,dc=ldap
tlsConfigs:
  enabled: true
  secretRef: [***SURVEYOR TLS CERT SECRET***]
ingress:
 enabled: true
 protocol: HTTPS
  className: nginx
 rules:
    path: "/"
    host: "MY-DOMAIN.EXAMPLE.COM"
    port: 8443
  tls:
    enabled: true
    secretRef: "[***INGRESS TLS CERT SECRET***]"
globalTruststore:
  secretRef:
    name: [***LDAP TRUSTSTORE SECRET***]
    key: [***TRUSTSTORE SECRET KEY***]
  type: PKCS12
 password:
    secretRef:
      name: [***LDAP TRUSTSTORE SECRET***]
      key: [***TRUSTSTORE PW SECRET KEY***]
```

For clusterConfigs

clusterConfigs specifies the Kafka clusters that Cloudera Surveyor connects to. Clusters specified here are the ones that will be available on the UI for monitoring and management.

- clusterConfigs.clusters[*] An array of Kafka clusters and their configuration. Each entry defines the configuration for a single Kafka cluster.
- clusterConfigs.clusters[*].clustername The name of the cluster. This name is displayed on the UI.
- clusterConfigs.clusters[*].bootstrapServers A comma-separated list of the bootstrap servers for the Kafka cluster that Cloudera Surveyor connects to. Specify multiple servers for highly available connections.
- clusterConfigs.clusters[*].tags User defined tags. Used for organization and filtering.
- clusterConfigs.clusters[*].commonClientConfig Kafka client configuration properties applied to all clients for this cluster. Must contain upstream Kafka client properties as a map. The exact properties that

you specify here depend on the security configuration of the Kafka cluster that you want to connect. This example specifies a Kafka cluster that uses PLAIN authentication with TLS.

Sensitive property values are referenced from Secrets instead of being hardcoded. Secrets containing sensitive properties are mounted using the secretsToMount property. References use Kafka DirectoryConfigProvider syntax. For more information, see *Registering Kafka clusters*.

• clusterConfigs.clusters[*].authorization.enabled – Enables authorization for this cluster.

For secretsToMount

secretsToMount specifies the Secrets to mount in the Cloudera Surveyor Container. You use secretsT oMount to mount the sensitive values required for Kafka client configuration specified in clusterConfigs.

- secretsToMount[*].create Specifies whether to create the Secret. Set to false in this example as the Secrets are assumed to already exist.
- secretsToMount[*].secretRef The name of the Secret to mount.
- secretsToMount[*].items[*].key The key in the Secret to mount.
- secretsToMount[*].items[*].path The path where the item is mounted. The path is relative to /opt/ secrets/[***SECRET NAME***]/ in the Cloudera Surveyor Container.

For surveyorConfig

surveyorConfig specifies global configuration for Cloudera Surveyor. This example sets various authentication properties.

- surveyorConfig.surveyor.authentication.enabled Enables or disables authentication. Set to true by default. Included in the example as a reference, you do not need to set the property explicitly to enable authentication.
- surveyorconfig.surveyor.authentication.* These properties control user session behavior and authentication token management. They define the session timeout, inactivity timeout, and the interval for renewing authentication tokens. All time intervals are specified in ISO 8601 format. For optimal security, Cloudera recommends that token renewal is shorter than or equal to half of the inactivity timeout.
- surveyorconfig.quarkus.security.ldap.dircontext.* These properties configure the LDAP server that Cloudera Surveyor connects to. They specify the server URL, the distinguished name (DN) of the bind user, and the password of the bind user. These are required for establishing a secure connection with the LDAP directory.



Important: To ensure that Cloudera Surveyor connects to the LDAP server securely, the URL you specify in surveyorconfig.quarkus.security.ldap.dir-context.url must start with ldaps://.

 surveyorconfig.quarkus.security.ldap.identity-mapping.* - These properties configure how Cloudera Surveyor interacts with the LDAP directory to identify users and map their group memberships. They define the attributes and base DNs used to locate user entries and groups in the directory, as well as the filters applied to verify group membership.



Note: Groups are not required for authentication. You might want to configure the surveyorConf ig.quarkus.security.ldap.identity-mapping.attribute-mappings.* properties accordingly. For more information on the surveyorConfig.quarkus.* properties, see *Using Security with an LDAP Realm* in the Quarkus documentation.

For tlsConfigs

tlsConfigs enables TLS and specifies the Secret containing the certificate of Cloudera Surveyor.

• tlsConfigs.enabled – Enables or disables TLS. Set to true by default. Included in the example as a reference, you do not need to set the property explicitly to enable TLS.

• tlsConfigs.secretRef – Name of the Secret containing the Cloudera Surveyor certificate and key.

For ingress

ingress enables the creation of an Ingress. The Ingress provides secure external access to the Cloudera Surveyor UI.

- ingress.enabled Enables or disables Ingress.
- ingress.protocol Configures the Ingress protocol.
- ingress.className The class name of the Ingress controller. This example configures the Ingress-Nginx controller.
- ingress.rules.host Specifies the DNS hostname that the Ingress controller should match for incoming HTTP/HTTPS requests.
- ingress.rules.port The port of the Ingress rule. This is the port of the Kubernetes Service that the Ingress forwards requests to.
- ingress.tls.enabled Enables TLS for the Ingress.
- ingress.tls.secretRef The name of the Secret that contains Ingress TLS certificates.



Note: If you are using cert-manager, configure ingress.tls.issuer and set it to the name of your Issuer resource. If set, Ingress will automatically request a certificate and save it to the Secret specified in ingress.tls.secretRef.

For globalTruststore

globalTruststore specifies the Secrets containing the truststore of the LDAP server and the password for the truststore.

- globalTruststore.secretRef.name The name of the Kubernetes Secret containing the truststore of the LDAP server.
- globalTruststore.secretRef.key The key in the Kubernetes Secret that contains the truststore.
- globalTruststore.password.name The name of the Kubernetes Secret containing the truststore password.
- globalTruststore.password.key The key in the Kubernetes Secret that contains the truststore password.

11. Install Cloudera Surveyor with helm install.

```
helm install cloudera-surveyor \
    --namespace [***NAMESPACE***] \
    --values [***VALUES FILE***] \
    --set-file surveyorConfig.surveyor.authentication.keys.active=[***PATH
    TO AUTHENTICATION KEY FILE***] \
    --set image.registry=[***REGISTRY HOSTNAME***]:[***PORT***] \
    --set 'image.imagePullSecrets=[***REGISTRY CREDENTIALS SECRET***]' \
    --set-file clouderaLicense.fileContent=[***PATH TO LICENSE FILE***] \
    oci://[***YOUR REGISTRY***]/cloudera-helm/csm-operator/surveyor \
    --version 1.4.0-b199
```

- The string cloudera-surveyor is the Helm release name of the chart installation. This is an arbitrary, user defined name. Cloudera recommends that you use a unique and easily identifiable name.
- *[***VALUES FILE***]* is the values file you prepared in Step 10 on page 22.
- surveyorConfig.surveyor.authentication.keys.active specifies the file containing the authentication key that you generated in Step 9 on page 22. The key is required for LDAP authentication to function.
- imagePullSecrets specifies what Secret is used to pull images from the specified registry. Ensure that you replace [***REGISTRY CREDENTIALS SECRET***] with the name of the Secret you created in Step 4 on page 21.
- clouderaLicense.fileContent is used to register your license. If this property is set, a Secret is generated that contains the license you specify. Setting this property is mandatory. Cloudera Surveyor will not function

without a valid license. Ensure that you replace [***PATH TO LICENSE FILE***] with the full path to your Cloudera license file.

- You can use --set to override properties that are defined in your values file, or add additional properties that are not present in your values file.
- 12. Verify your installation.

This is done by listing the Deployments and Pods in your namespace. If installation is successful, a Cloudera Surveyor Deployment and two Pods will be present in the cluster.

```
kubectl get deployments --namespace [***NAMESPACE***]
```

NAME		READY	UP-TO-DATE	AVAILABLE	AGE
#	2 (2	0	2	1 2	
cloudera-surveyor	2/2	2	2	⊥3m	

kubectl get pods --namespace [***NAMESPACE***]

NAME #	READY	STATUS	RESTARTS	AGE	
cloudera-surveyor-649f755f	6d-689gx	1/1	Running		0
cloudera-surveyor-649f755f	6d-xj2kp	1/1	Running		0

13. Access the Cloudera Surveyor UI.

The UI is accessible by connecting to the Ingress.

```
kubectl get ingress cloudera-surveyor-ingress --namespac
e [***NAMESPACE***]
```

```
NAME CLASS HOSTS ADDRESS
PORTS
cloudera-surveyor-ingress nginx my-domain.example.com 10.14.91.1 80,
443
```

Typically you will be able to access the UI through the host and port listed. However, the exact port that you have to use might be infrastructure dependent. If you are unable to connect, check the configuration of your Ingress controller. You can also try connecting directly to the Service of the Ingress.

Results

Cloudera Surveyor is installed. You can now manage and monitor your Kafka clusters using the UI. Related Information Registering Kafka clusters Using Security with an LDAP Realm | Quarkus

Installing Cloudera Surveyor for evaluation

Complete these steps to install a basic deployment of Cloudera Surveyor that has no security configured. Use these instructions if you want to install quickly in a development environment for proof of concept or evaluation purposes.

Before you begin

- Ensure that your Kubernetes environment meets requirements listed in System requirements.
- Your Kubernetes cluster requires internet connectivity to complete these steps. It must be able to reach the Cloudera Docker registry.

- 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 installation artifacts are hosted.
- Ensure that you have access to a valid Cloudera license.
- Review the Helm chart reference before installation.

The Helm chart accepts various configuration properties that you can set during installation. Using these properties you can customize your installation.

Procedure

1. Create a namespace in your Kubernetes cluster.

```
kubectl create namespace [***NAMESPACE***]
```

This is the namespace where you install Cloudera Surveyor. Use the namespace you create in all installation steps that follow.

2. Prepare a Docker configuration JSON file that includes your Cloudera credentials.

```
{
   "auths": {
    "container.repository.cloudera.com": {
        "auth": "[***BASE64 ENCODED USERNAME:PASSWORD***]"
     }
}
```

The value of auth is the base64-encoded string of your credentials, with your username and password separated by a colon (:). For example, if your username is myuser and your password is pass, you must encode myuser:pass.

3. Create a Kubernetes Secret containing your Cloudera credentials.

```
kubectl create secret docker-registry [***CLOUDERA CREDENTIALS SECRET***]
\
--from-file=.dockerconfigjson=[***PATH TO CLOUDERA CREDENTIALS JSON***]
--namespace=[***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. Prepare a custom values file (values.yaml).

The values file contains configuration for Cloudera Surveyor. This file specifies the Kafka clusters that Cloudera Surveyor connects to as well as various other configuration properties.

```
clusterConfigs:
  clusters:
    - clusterName: [***CLUSTER NAME***]
    tags:
        - [***TAG1***]
        - [***TAG2***]
    bootstrapServers: [***BOOTSTRAP SERVERS***]
    adminOperationTimeout: PT1M
    authorization:
        enabled: false
        commonClientConfig:
            security.protocol: PLAINTEXT
surveyorConfig:
        surveyor:
```

AGE

```
authentication:
enabled: false
tlsConfigs:
enabled: false
```

- clusterConfigs.clusters[*] An array of Kafka clusters and their configuration. Each entry defines the configuration for a single Kafka cluster.
- clusterConfigs.clusters[*].clustername The name of the cluster. This name is displayed on the UI.
- clusterConfigs.clusters[*].bootstrapServers A comma-separated list of the bootstrap servers for the Kafka cluster that Cloudera Surveyor connects to. Specify multiple servers for highly available connections.
- clusterConfigs.clusters[*].commonClientConfig Kafka client configuration properties applied to all clients for this cluster. Must contain upstream Kafka client properties as a map. The exact properties that you specify here depend on the security configuration of the Kafka cluster that you want to connect. This example specifies a Kafka cluster that is unsecure. For more information, see *Registering Kafka clusters*.
- All security-related properties are set false to disable security. These properties must be explicitly set to false as the default value for all of them is true.
- 6. Install Cloudera Surveyor with helm install.

```
helm install cloudera-surveyor \
    --namespace [***NAMESPACE***] \
    --values [***VALUES FILE***] \
    --set 'image.imagePullSecrets=[***CLOUDERA CREDENTIALS SECRET***]' \
    --set-file clouderaLicense.fileContent=[***PATH TO LICENSE FILE***] \
    oci://container.repository.cloudera.com/cloudera-helm/csm-operator/surv
eyor \
    --version 1.4.0-b199
```

- The string cloudera-surveyor is the Helm release name of the chart installation. This is an arbitrary, user defined name. Cloudera recommends that you use a unique and easily identifiable name.
- [***VALUES FILE***] is the values file you prepared in Step 5 on page 27.
- imagePullSecrets specifies what secret is used to pull images from the Cloudera registry. Setting this property is mandatory, otherwise, Helm cannot pull the necessary images from the Cloudera Docker registry. Ensure that you replace [***CLOUDERA CREDENTIALS SECRET***] with the name of the Secret you created in Step 3 on page 27.
- clouderaLicense.fileContent is used to register your license. If this property is set, a Secret is generated that contains the license you specify. Setting this property is mandatory. Cloudera Surveyor will not function without a valid license. Ensure that you replace [***PATH TO LICENSE FILE***] with the full path to your Cloudera license file.
- You can use --set to override properties that are defined in your values file, or add additional properties that are not present in your values file.
- 7. Verify your installation.

This is done by listing the Deployments and Pods in your namespace. If installation is successful, a Cloudera Surveyor Deployment and two Pods will be present in the cluster.

```
kubectl get deployments --namespace [***NAMESPACE***]
NAME READY UP-TO-DATE AVAILABLE
#...
cloudera-surveyor 2/2 2 2 13m
```

```
kubectl get pods --namespace [***NAMESPACE***]
```

```
NAME READY STATUS RESTARTS AGE #...
```

aloudone announce 64057556d 600an	1 / 1		0
cloudera-surveyor-6491/5516d-689gx	\perp / \perp	Running	0
13m			
cloudera-surveyor-649f755f6d-xj2kp	1/1	Running	0
13m			

8. Access the Cloudera Surveyor UI.

Installation by default creates a NodePort type Service for Cloudera Surveyor. The UI is accessible from any of the Kubernetes cluster nodes on the external port of the Service. List Services to get the external port.

```
kubectl get service cloudera-surveyor-service --namespac
e [***NAMESPACE***]
```

NAME		TYPE	CLUSTER-IP	EXTERNAL-IP	PORT
(S)	AGE				
cloudera-surv	veyor-service	NodePort	10.43.196.52	<none></none>	808
0:30525/TCP	16m				

In this example, the external port is 30525.

Results

Cloudera Surveyor is installed. You can now manage and monitor your Kafka clusters using the UI. Related Information Registering Kafka clusters

Installing from OperatorHub in OpenShift

Learn how to install Cloudera Streams Messaging - Kubernetes Operator from OperatorHub in OpenShift.

About this task



Important: When installing from OperatorHub, Cloudera Streams Messaging - Kubernetes Operator is installed using Operator Lifecycle Manager (OLM). Customizing your installation and setting the properties of the Strimzi Cluster Operator is limited both during and following installation.

Installation from OperatorHub in Openshift involves creating two Secrets in your installation namespace. One containing your Cloudera license, and one containing your Cloudera credentials (username and password). The license is required for Cloudera Streams Messaging - Kubernetes Operator to function properly. The credentials provide access to the Cloudera Docker registry (container.repository.cloudera.com) where installation artifacts are pulled from.

After the Secrets are available in your cluster, you can continue with the standard process of installing operators from OperatorHub.

Before you begin

- Ensure that you have access to your Cloudera credentials (username and password).
- Ensure that you have access to a valid Cloudera license.
- These instructions use oc to create Secrets. However, you can also create both Secrets using the OpenShift web console.

Procedure

1. Create a Secret containing your license.

The name of the Secret is fixed. It must be called csm-op-license.

```
oc create secret generic csm-op-license --from-file=license.txt=[***PATH
   TO LICENSE FILE***]
```

2. Create a Secret containing your Cloudera credentials.

The name of the Secret is fixed. It must be called cloudera-container-repository-credentials.



Note: The Secret containing your Cloudera credentials must also be available in all namespaces where you deploy Kafka or Kafka Connect clusters. Cloudera recommends that you create the Secret in all required namespaces now if you know what namespaces you will be using to deploy Kafka or Kafka Connect.

3. Install Cloudera Streams Messaging - Kubernetes Operator from OperatorHub using the web console or the CLI. For detailed steps, see Adding Operators to a cluster in the OpenShift documentation.

What to do next

- Deploy a Kafka cluster, see Deploying Kafka.
- Set up Prometheus for monitoring, see Configuring Kafka for Prometheus monitoring and Monitoring with Prometheus.