Cloudera Streams Messaging Operator for Kubernetes 1.6.0

Overview

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

https://docs.cloudera.com/

Legal Notice

Contents

Deployment architecture

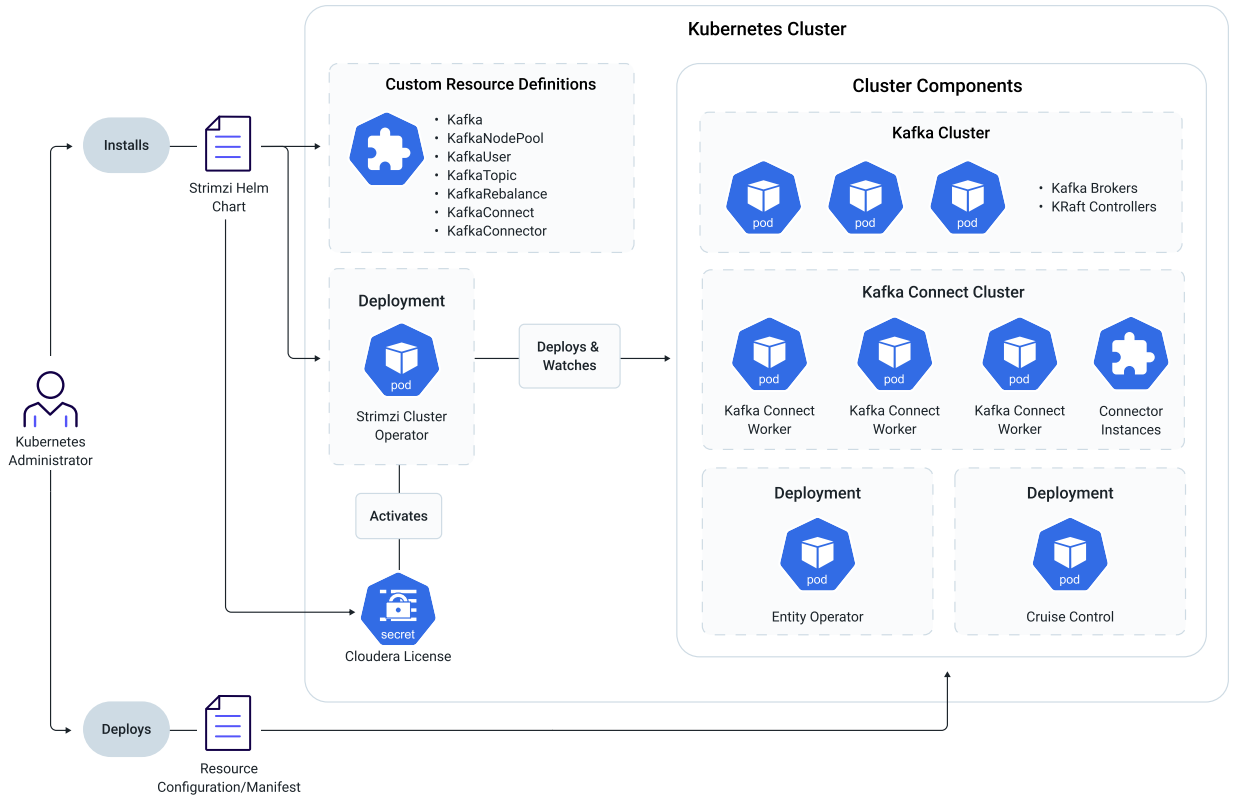
Learn the architecture of typical Cloudera Streams Messaging Operator for Kubernetes deployments.

Cloudera Streams Messaging Operator for Kubernetes includes two components that you can install. Strimzi and Cloudera Surveyor. 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.

The two components are independent. You can choose to install only Strimzi or only Cloudera Streams Messaging Operator for Kubernetes. Whether you choose to install one or both components depends on your use case and operational objectives. Cloudera recommends installing both.

Strimzi

The following is the typical architecture of a Strimzi and Kafka deployment in Cloudera Streams Messaging Operator for Kubernetes.



Strimzi is installed with a Helm chart. The Helm chart installs various Custom Resource Definitions (CRDs) to the Kubernetes environment. Resources described by these CRDs are managed by the Kubernetes operator applications. Strimzi defines CRDs such as:

- `Kafka` – represents a Kafka cluster consisting of Kafka brokers, KRaft controllers, Cruise Control and Strimzi Entity Operators.
- `KafkaNodePool` – represents a group of Kafka brokers from the cluster that have the same configuration.
- `KafkaTopic` – represents a Kafka topic.
- `KafkaUser` – represents a user that is external to the Kafka cluster.
- `KafkaRebalance` – represents a broker rebalance action for the Kafka cluster.
- `KafkaConnect` – represents a Kafka Connect cluster consisting of one or more Kafka Connect workers.

- `KafkaConnector` – represents a Kafka Connect connector instance.

Note: Strimzi includes CRDs not highlighted here. The use of these CRDs are either not supported in Cloudera Streams Messaging Operator for Kubernetes, or are for internal use by the Strimzi Cluster Operator.

When installing the Helm chart, a Strimzi Cluster Operator Kubernetes `Deployment` is created with a Strimzi Cluster Operator `Pod` running within the deployment. This application is responsible for monitoring cluster components and reconciling these components when their configuration changes. During installation you are required to register a license, which activates the Strimzi Cluster Operator.

Following installation, you deploy various custom resources in the cluster like `Kafka` and `KafkaNodePool` resources. Based on the configuration in the resource, the Strimzi Cluster Operator deploys clusters of the components described by the resources.

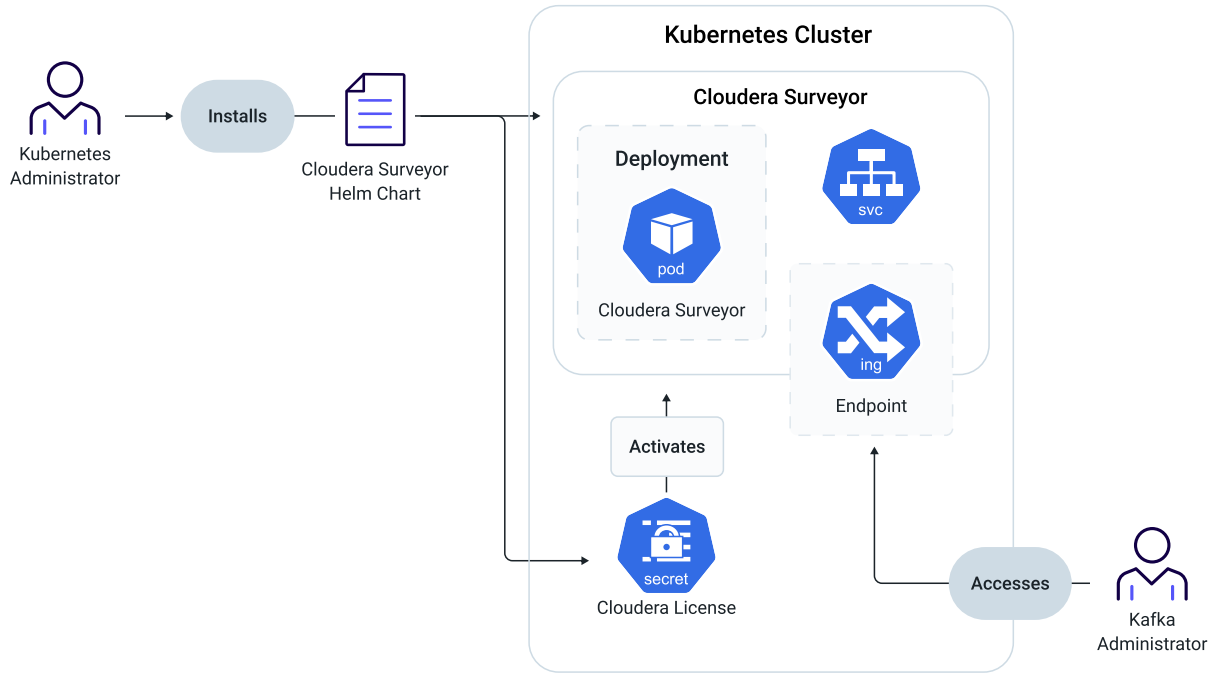
Specifically, `Kafka` and `KafkaNodePool` resources will deploy Kafka clusters. Optionally, if configured, the `Kafka` resource also deploys the Strimzi Entity Operator and Cruise Control. The Strimzi Entity Operator is responsible for managing other resources inside the particular Kafka cluster (topics, users, and so on), Cruise Control is used for rebalancing Kafka.

`KafkaConnect` and `KafkaConnector` resources are used to deploy Kafka Connect clusters and instances of Kafka Connect connectors.

Warning: Strimzi allows creating Kafka brokers by creating only a single `Kafka` resource. However, Cloudera Streams Messaging Operator for Kubernetes only supports creating Kafka brokers by creating `KafkaNodePool` resources. Node pools allow for more flexible deployments with easier scaling options. Moreover, certain features like rack awareness and scaling are limited without node pools. Broker creation using the `Kafka` resource only is deprecated, and results in unnecessary effort of migrating the deployment to use node pools.

Cloudera Surveyor

The following is the typical architecture of a Cloudera Surveyor deployment in Cloudera Streams Messaging Operator for Kubernetes.

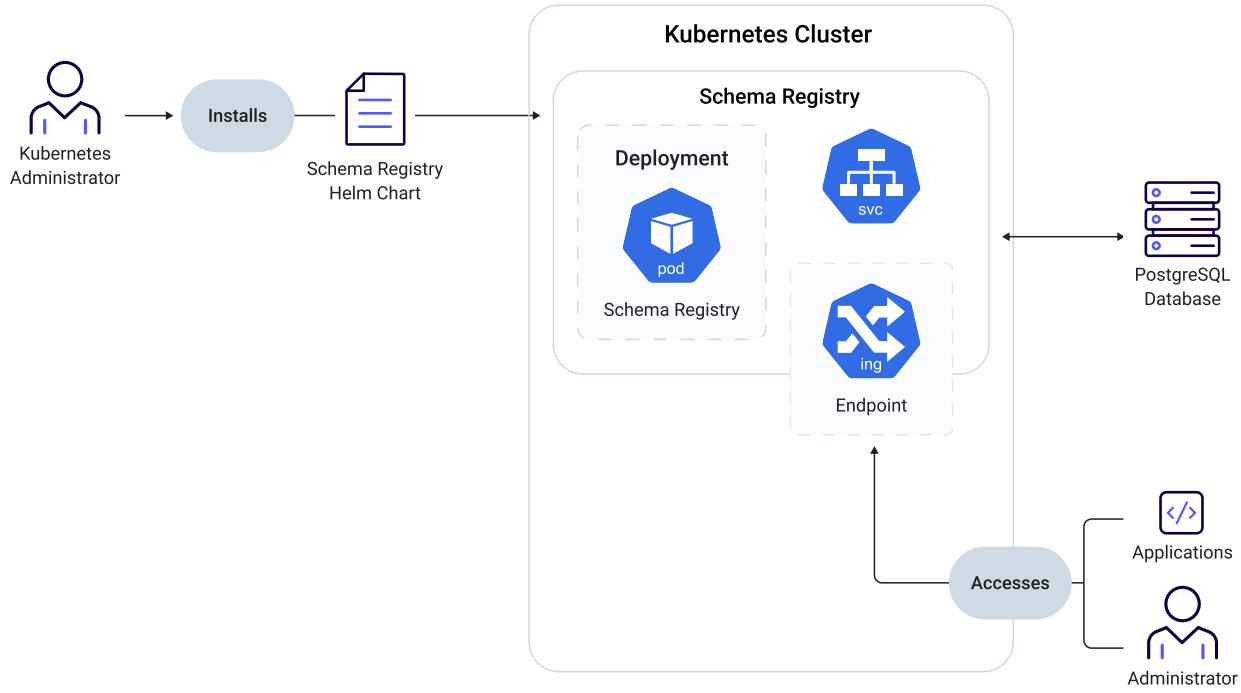


Cloudera Surveyor is installed with a Helm chart. The Helm chart creates a Cloudera Surveyor `Deployment` and `Service`. Kubernetes creates the associated Cloudera Surveyor `Pods` as a result of the `Deployment`. The replica

count for the `Deployment` is set to two by default, resulting in the creation of two `Pods` for high availability. Optionally, an `Ingress` is deployed as well if configured. Users access the Cloudera Surveyor UI through the `Ingress` and `Service`.

Schema Registry

The following is the typical architecture of a Schema Registry deployment in Cloudera Streams Messaging Operator for Kubernetes.



Schema Registry is installed with a Helm chart. The Helm chart creates a Schema Registry `Deployment` and `Service`. Kubernetes creates the associated Schema Registry `Pods` as a result of the `Deployment`. The replica count for the `Deployment` is set to two by default, resulting in the creation of two `Pods` for high availability. Optionally, an `Ingress` is deployed as well if configured.

Administrators and applications can access Schema Registry's UI and REST API through the `Ingress` and the `Service`.

Schema Registry uses a PostgreSQL database to persistently store schema definitions and metadata. For testing and evaluation purposes, an in-memory database option is available, but it is not recommended for production use as all schemas will be lost when `Pods` restart.

Related Information

What is Cloudera Surveyor?

Overview | Strimzi

Licensing

Cloudera Streams Messaging Operator for Kubernetes requires a valid license to function. Licenses are made available to you together with your Cloudera credentials as part of your license and subscription agreement with Cloudera.

Licenses are registered during Cloudera Streams Messaging Operator for Kubernetes installation. Specifically, during the installation of Strimzi. The license activates the Strimzi Cluster Operator. Licenses are stored in a Kubernetes secret. Licenses can be updated at any time.

Licenses are valid for a set period of time. Once the license expires, the cluster resources you deployed will continue to run. However, reconciliation of resources will be blocked. For example, failed pods will not be restarted, scaling your clusters will not be possible. In general, the control mechanisms in place that keep resources healthy will be blocked. This leads to deployed resources breaking down over time.

Cloudera Streams Messaging Operator for Kubernetes publishes various log entries and Kubernetes events related to your licenses.

For example, if your license expires or becomes invalid due to any reason, appropriate logs and events are published notifying you that there are issues with your license.

These logs and events are published for the Strimzi Cluster Operator deployment. You can check these logs and events with the following commands.

```
kubectl events deployments/strimzi-cluster-operator --namesp
ace [***NAMESPACE***]
```

```
kubectl logs deployment/strimzi-cluster-operator --names
pace [***NAMESPACE***]
```

Related Information

Updating a license

Sizing and performance considerations

Learn about ways you can size your deployment for optimal performance.

Kafka broker performance primarily depends on the IO bandwidth of the nodes and disks. Because of this, Cloudera recommends using SSDs with high IOPS and throughput for large workloads. JBOD can also lead to throughput improvements when the node IO bandwidth can support multiple disks.

When optimizing for large workloads, using HDDs and storage replication services such as Longhorn might add a significant performance overhead.

Depending on the characteristics of the workload, brokers might require a large memory pool to be able to serve fetch requests from the cache. Brokers might also require an increased CPU allocation to support compressed messages

KRaft Controllers require a small resource pool in most workloads, and scaling the cluster to more than three nodes usually provides no benefit.

Recommended minimum setup

Cloudera recommends the following cluster sizing as a baseline for small and medium workloads.

Container	Count	CPU (m) per Pod	Memory (MiB) per Pod	Notes
Cloudera Surveyor	2	4000	20480	Required for Cloudera Surveyor. Memory requirements heavily depend on the number of Kafka clusters that Cloudera Surveyor monitors and manages.
Schema Registry	2	1000	2048	Required for Schema Registry.

Container	Count	CPU (m) per Pod	Memory (MiB) per Pod	Notes
Strimzi Cluster Operator	2	1000	384	Required.
Kafka Broker	3	8000	20480	Required for Kafka workloads.
KRaft Controller	3	4000	4096	Required for Kafka workloads.
Cruise Control	1	4000	4096	Required for rebalance operations.
Topic Operator	1	500	256	Required if you want to manage topics with `KafkaTopic` resources.
User Operator	1	500	256	Required if you want to manage Kafka users with `KafkaUser` resources.
Kafka Exporter	1	500	256	Required if you want to have additional broker and client metrics available.
Kafka Connect	3	4000	4096	Required if you want to use Kafka Connect and related functionality.

FIPS mode in Cloudera Streams Messaging Operator for Kubernetes

You can deploy Cloudera Streams Messaging Operator for Kubernetes components in FIPS 140-2 compliant mode to meet strict security requirements. Strimzi and its managed components automatically enable FIPS mode on FIPS-enabled RHEL 8.x or higher hosts. Cloudera Surveyor for Apache Kafka is FIPS compliant, but requires you to manually enable FIPS mode through configuration. Cloudera Surveyor also supports the use of custom security providers.

Host prerequisites

Cloudera Streams Messaging Operator for Kubernetes requires host prerequisites to run in FIPS mode.

The following host prerequisites are valid for Cloudera Streams Messaging Operator for Kubernetes:

- Host operating system must be Red Hat Enterprise Linux (RHEL) 8.x or a higher version.
- FIPS mode must be enabled on the host.

If the host is FIPS-enabled, `Containers` running on that host will be FIPS-enabled too. All container images shipped with Cloudera Streams Messaging Operator for Kubernetes are built using RHEL9-based Universal Base Images (UBI) that are FIPS-compliant. This means that running `Containers` will be RHEL-based and FIPS-compliant.

Strimzi

Strimzi as well as components managed by Strimzi are FIPS 140-2 compliant by default and run in FIPS mode automatically in FIPS-enabled environments.

When deployed on a FIPS-enabled cluster, the OpenJDK in the `Containers` of Strimzi automatically activates its FIPS mode. This requires no explicit configuration for Strimzi, it automatically uses the FIPS-compliant cryptographic libraries built into its Java runtime. Inserting additional security providers is not required.

Strimzi is also responsible for managing Kafka and Cruise Control. When deployed in FIPS mode, Strimzi generates keystores, truststores, and configuration files for these components in a FIPS-compatible way. Running Kafka or Cruise Control in FIPS mode does not require any explicit configuration, and additional security providers are not required.

Note:

FIPS can be disabled by setting the FIPS_MODE environment variable of Strimzi Cluster Operator `Deployment` to disabled. When FIPS is disabled, Strimzi and all managed components run the same as they would run on a non#FIPS Kubernetes cluster.

Related Information

Disabling FIPS mode using Cluster Operator configuration | Strimzi

Cloudera Surveyor

Cloudera Surveyor supports running in a FIPS 140-2 compliant manner. However, FIPS mode is disabled by default and must be explicitly enabled.

Enabling FIPS mode enforces strict security controls and changes the configuration of Cloudera Surveyor in the following ways:

- The Java runtime is configured to use a FIPS#capable cryptographic provider first so that cryptographic operations use FIPS#approved implementations.
- All TLS truststores must be in the BCFKS (Bouncy Castle FIPS Keystore) format and must be password-protected.
- The security settings of the Java Virtual Machine are adjusted so the runtime uses FIPS#approved algorithms and a cryptographically strong source of randomness.

If required, you can also configure the use of additional, custom security providers. These must be mounted to the Cloudera Surveyor `Container`.

Enabling FIPS mode for Cloudera Surveyor

Enable FIPS mode for Cloudera Surveyor by setting the fipsMode property to true in your values file.

```
#...
fipsMode: true
```

Configuring additional FIPS security providers in Cloudera Surveyor

Cloudera Surveyor integrates and uses a FIPS-compliant security provider when FIPS mode is enabled. If required, you can mount additional custom security providers and configure Cloudera Surveyor to use them.

Before you begin

- A volume that contains necessary provider JARs must be available. The volume should support RWX.
- The following steps use CryptoComply for Java (CCJ) SafeLogic - Java JCE Provider as an example.

Procedure

1. Mount the provider JAR file to the Cloudera Surveyor `Container`.

   Use the extraVolumes and extraVolumeMounts properties in your values file.

   ```
   #...
   extraVolumes:
     - name: [***EXTRA VOLUME NAME***]
       persistentVolumeClaim:
         claimName: [***RWX VOLUME CLAIM***]
   ```

```
extraVolumeMounts:
  - name: [***EXTRA VOLUME NAME***]
    mountPath: /opt/providers
    readOnly: false
```

- extraVolumes – List of additional volumes to attach to the `Pod`. This allows you to mount various types of volumes, such as secret, configMap, or persistentVolumeClaim.
- extraVolumes[*].name – The name of the volume. Must match the corresponding name in the extraVolumeMounts property.
- extraVolumes[*].persistentVolumeClaim – The volume configuration. Can be any valid Kubernetes volume type, such as secret, configMap, or persistentVolumeClaim.
- extraVolumeMounts – List of volume mounts that specify how volumes are mounted into the `Container`.
- extraVolumeMount[*].name – The name of the volume to mount. Must match a volume defined in the extraVolumes property.
- extraVolumeMount[*].mountPath – The path within the `Container` where the volume is mounted.
- extraVolumeMount[*].readOnly – Whether the volume is mounted read-only. The default value is false.

2. Configure Cloudera Surveyor to use the provider.

   Use the securityProviders property in your values file. This appends your custom provider class to the list of security providers.

```
#...
securityProviders:
  - providerClassName: "com.safelogic.cryptocomply.jcajce.provider.CryptoComplyFipsProvider"
    argument: "[***PROVIDER ARGUMENT***]"
    classpath:
      - "/opt/providers/ccj-3.0.2.1.jar"
```

- securityProviders – List of security providers to load.
- securityProviders[*].providerClassName – Class name of the provider.
- securityProviders[*].argument – Argument to pass to provider.
- securityProviders[*].classpath – List of filesystem paths to add to the classpath for this provider to function. This is an absolute file path in the /my-dir/my-file form, in which my-dir is the mountPath from the extraVolumeMounts property and my-file is the provider file name in that directory.

Schema Registry

Schema Registry is FIPS 140-2 compliant by default and runs in FIPS mode automatically in FIPS-enabled environments. Explicit configuration is not required to enable FIPS mode.

Limitations

FIPS mode in Cloudera Streams Messaging Operator for Kubernetes has limitations for Strimzi and Kafka.

Strimzi

JMX monitoring has the following limitations in FIPS mode that make it unsuitable for secure monitoring:

- **JMX authentication** – Password-based authentication for JMX is not supported in FIPS mode. This forces you to expose JMX without authentication, creating significant security risks.
- **JMX Trans compatibility** – The JMX Trans tool is outdated and does not support Java 17. The Strimzi container image for JMX Trans uses Java 11, which lacks the improved FIPS support found in modern OpenJDK versions, creating compatibility and security issues.

For these reasons, Cloudera recommends using Prometheus metrics instead of JMX for monitoring Strimzi in FIPS environments.

Kafka

The following FIPS mode limitations are valid for Kafka:

- For systems employing Simple Authentication and Security Layer (SASL), Cloudera recommends explicitly select a strong mechanism such as SCRAM-SHA-512 to ensure FIPS compatibility by default.
- In environments where required security algorithms are unavailable, some SASL mechanisms, such as PLAIN, as selected authentication might fail. Although client-side configurations can sometimes serve as a workaround, Cloudera does not recommend client-side configuration workarounds. Client-side modifications might introduce non-compliant cryptographic modules or policies, which could compromise the FIPS compatibility of the entire cluster. Always use a fully FIPS-compliant mechanism instead.