Cloudera Streams Messaging Operator for Kubernetes 1.6.0

Cloudera Surveyor for Apache Kafka Security

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

https://docs.cloudera.com/

Legal Notice

Contents

Channel encryption (TLS) in Cloudera Surveyor for Apache Kafka

Get started with TLS encryption in Cloudera Surveyor. Learn which types of traffic can be secured using TLS, along with the available configuration properties and certificate management practices.

You can protect sensitive data and ensure secure access by configuring TLS for Cloudera Surveyor. TLS can be configured for the following types of traffic.

- **TLS for internal (backend) traffic** – Secures traffic between the external access point (`LoadBalancer` or `Ingress`) and Cloudera Surveyor. Internal TLS ensures that even internal traffic within the Kubernetes cluster remains encrypted, protecting against potential threats inside the cluster network.
- **TLS for client traffic** – Secures connections from end users (clients) to the Cloudera Surveyor UI. Configured on the Kubernetes `Service` that provides external access (`LoadBalancer` or `Ingress`). Ensures that all data exchanged with clients connecting from the outside world are encrypted.
- **TLS for LDAP traffic** – Secures connections between Cloudera Surveyor and your LDAP authentication provider. Typically, you set this up when you configure LDAP authentication.

Cloudera recommends that you enable TLS for all of the above listed categories. This ensures that all traffic handled by Cloudera Surveyor is secure.

Configuring TLS channel encryption

Configure TLS with TLS-related properties in your custom values file. TLS for internal, client, and LDAP traffic is configured separately.

TLS for internal traffic

TLS for internal traffic is configured with the tlsConfigs.* properties. For example:

```
tlsConfigs:
  enabled: true
  secretRef: "[***TLS SECRET NAME***]"
```

- tlsConfigs.enabled – Enables or disables TLS
- tlsConfigs.secretRef – The name of the Kubernetes `Secret` containing the TLS certificate and private key. This `Secret` must be created in advance or managed by cert-manager.

TLS for client traffic

TLS for client traffic is configured on the `Service` (`Ingress` or `LoadBalancer`) that provides access to Cloudera Surveyor.

For `Ingress`, you configure ingress.tls.* properties. For example:

```
#...
ingress:
  enabled: true
  protocol: "HTTPS"
  className: "nginx"
  rules:
    host: MY-DOMAIN.EXAMPLE
    port: 443
  tls:
    enabled: true
    issuer: "[***ISSUER NAME***]"
```

```
    secretRef: "[***INGRESS TLS CERT SECRET***]"
```

TLS for `Ingress` is typically configured when you set up external access with `Ingress`. For a step-by-step guide see *Configuring external access with Ingress*.

For `LoadBalancers`, the actual load balancer is provisioned and managed by your cloud or infrastructure provider. As a result, TLS settings and certificate management may vary depending on the platform. Refer to vendor-specific documentation for detailed guidance on configuring TLS.

TLS for LDAP traffic

TLS for LDAP traffic is configured using globalTruststore.* properties. These properties specify a `Secret` that contains a truststore with the TLS certificate of the root Certificate Authority (CA) of the LDAP certificate chain as well as a password to access the truststore. You typically configure these properties when you set up LDAP. For a step-by-step guide, see *Configuring LDAP authentication*

Related Information

Configuring external access with Ingress

Configuring LDAP authentication

Managing certificates

Learn about managing TLS certificates in Cloudera Surveyor. You can manage certificates manually or use cert-manager to automate certificate management. Cloudera recommends automatic certificate management.

Cloudera Surveyor stores its TLS certificates in `Secrets`. The `Secrets` are specified by the following two properties.

- tlsConfigs.secretRef – Specifies the name of the `Secret` containing the TLS certificate used for internal TLS traffic.
- ingress.tls.secretRef – Specifies the name of the `Secret` containing the TLS certificate used by the `Ingress` to secure incoming client communications.

Both `Secrets` must contain a valid certificate and private key. Cloudera recommends that you use cert-manager to manage the `Secrets` and the certificates that they store. Alternatively, you can choose to manage them manually.

Automatic certificate management with cert-manager

cert-manager is a popular Kubernetes add-on for automating the management and issuance of TLS certificates. In order to manage the certificates used by Cloudera Surveyor with cert-manager, you need the following.

- A cert-manager instance in your Kubernetes cluster.
- An `Issuer` deployed for cert-manager.

The management of the two `Secrets` that Cloudera Surveyor uses to store certificates differs.

- tlsConfigs.secretRef – The `Certificate` resource for this `Secret` must be created manually. When creating a `Certificate` resource, set spec.secretName to the same value as tlsConfigs.secretRef. This way cert-manager saves the certificate and private key to the `Secret` that Cloudera Surveyor uses.
- ingress.tls.secretRef – The `Certificate` resource for this `Secret` is created automatically when the cert-man ager.io/issuer: *[***ISSUER NAME***]* annotation is set in the ingress.extraAnnotations property.

  Specifically, the `Ingress` requests a certificate from cert-manager using `Issuer` name from the annotation. This triggers the creation of the `Certificate` resource and saves the certificate file and private key to the `Secret` defined in ingress.tls.secretRef.

Manual certificate management

When managing certificates manually, you must create the `Secrets` that contain the certificates and private keys manually. Ensure that you create and update the appropriate `Secrets` which are specified in tlsConfigs.secretRef and ingress.tls.secretRef.

Authentication in Cloudera Surveyor for Apache Kafka

Get started with authentication in Cloudera Surveyor. LDAP is the only supported authentication mechanism in Cloudera Surveyor.

Cloudera Surveyor supports authentication of its end users. It supports a single authentication method, which is LDAP. While configuring authentication is optional, it is enabled by default. As a result, you must configure LDAP authentication properties during or after installation.

Tip: If necessary, you can disable authentication by setting surveyorConfig.surveyor.authentication.enabled to false. However, Cloudera does not recommend you deploy Cloudera Surveyor with authentication disabled.

LDAP properties define the LDAP server Cloudera Surveyor uses to authenticate users. The LDAP server you configure must contain entries with valid usernames and passwords. When LDAP authentication is configured, users must provide valid credentials to access the Cloudera Surveyor UI.

Authentication keys and tokens

Cloudera Surveyor uses an authentication key to securely generate authentication tokens. These tokens are sent to clients and are required for subsequent access requests.

By default, Cloudera Surveyor automatically generates a cryptographically secure authentication key (128 random bytes) when authentication is enabled and no key is explicitly configured. However, you can also manually generate and configure the authentication key.

Manual configuration requires generating a sequence of at least 32 random bytes, saving it as a file, and configuring it using a configuration property. You can use any tool or method to generate the authentication key. Manually generating and configuring the authentication key is required in the following scenarios:

- When Cloudera Surveyor is deployed with FIPS mode enabled (fipsMode: true). Manual configuration is required to ensure proper auditability and compliance with FIPS standards.
- When you want to update an existing key.

The authentication key is sensitive data. It is stored in a Kubernetes `Secret` and mounted to all Cloudera Surveyor `Pods`. Because of this, Cloudera recommends following the security guidelines of your organization and restricting access to Cloudera Surveyor `Secrets` and `Pods`. Use standard Kubernetes access control mechanisms, such as Role-Based Access Control (RBAC), to ensure proper security.

When manually configuring the authentication key, Cloudera recommends the following:

- Delete the file containing the authentication key after configuration is complete.
- Update the authentication key on a regular basis. You can update the authentication key at any time through configuration.

Configuring LDAP authentication

Learn how to configure LDAP authentication in Cloudera Surveyor.

Before you begin

- TLS is enabled and configured for Cloudera Surveyor. See Channel encryption (TLS).

- An LDAP server is available that meets the following requirements:
  - The server has TLS enabled.
  - The server is accessible from the Kubernetes cluster where Cloudera Surveyor for Apache Kafka 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.

Important:

Credentials, including user passwords, provided by clients must be verified against what is available in LDAP. Sensitive data is passed from clients to Cloudera Surveyor and from Cloudera Surveyor to the LDAP server. Therefore, Cloudera strongly recommends that you configure TLS for both connections. TLS configuration between Cloudera Surveyor and the LDAP server is covered in the following steps. For TLS configuration between clients and Cloudera Surveyor, see Channel encryption (TLS).

Procedure

1. If deploying with FIPS mode enabled, manually generate an authentication key.

   When Cloudera Surveyor is deployed with FIPS mode enabled (fipsMode: true), you must manually generate and configure an authentication key.

   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.

   In non-FIPS deployments, you can skip this step. Cloudera Surveyor automatically generates a cryptographically secure random key if the key is not explicitly set.
2. 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
   ```
3. 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.
4. Create a `Secret` containing your LDAP principal and password (bind credentials).

   ```
   kubectl create secret generic [***LDAP CREDENTIALS SECRET***] \
     --namespace [***NAMESPACE***] \
     --from-literal=principal="$(echo -n 'Enter principal: ' >&2; read -s pr
   incipal; echo >&2; echo $principal)" --from-literal=password="$(echo -n
   'Enter password: ' >&2; read -s password; echo >&2; echo $password)"
   ```

   - Take note of *[***LDAP CREDENTIALS SECRET***]*. You will need to specify this name in a custom values file you create in a later step.
   - Enter your principal and the password for the principal when prompted. Example principal:

     ```
     cn=admin,dc=openldap-chart,dc=ldap
     ```

5. Configure LDAP properties in a custom values file (values.yml).

   The following configuration snippet is an example containing LDAP properties for a typical setup.

   

   Note: The surveyorConfig.surveyor.authentication.keys.active property is optional and is omitted from this example. When not configured, Cloudera Surveyor automatically generates a cryptographically secure random key.

   Explicitly configuring this property is only required when FIPS mode is enabled (fipsMode: tr ue). When configuring manually, Cloudera recommends that you set this property directly through the command line with the --set-file Helm option. An example is provided in a later step.

```
#...
surveyorConfig:
  surveyor:
    authentication:
      enabled: true
      userSessionTimeout: P1D
      inactivityTimeout: PT1H
      tokenRenewalInterval: PT10M
  quarkus:
    security:
      ldap:
        dir-context:
          url: ldaps://openldap-chart.ldap:1390
          principal: ${LDAP_PRINCIPAL}
          password: ${LDAP_PASSWORD}
        identity-mapping:
          rdn-identifier: uid
          search-base-dn: ou=users,dc=openldap-chart,dc=ldap
          attribute-mappings:
            "0":
              from: uid
              filter: (uid={0})
              filter-base-dn: ou=users,dc=openldap-chart,dc=ldap
env:
  - name: LDAP_PRINCIPAL
    valueFrom:
      secretKeyRef:
        name: [***LDAP CREDENTIALS SECRET***]
        key: principal
  - name: LDAP_PASSWORD
    valueFrom:
      secretKeyRef:
        name: [***LDAP CREDENTIALS SECRET***]
        key: password
globalTruststore:
  secretRef:
    name: [***LDAP TRUSTSTORE SECRET***]
    key: [***TRUSTSTORE SECRET KEY***]
  type: PKCS12
  password:
    secretRef:
      name: [***LDAP TRUSTSTORE SECRET***]
      key: [***TRUSTSTORE PW SECRET KEY***]
```

- 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.dir-context.* - 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.

  The bind user credentials (principal and password) are referenced from the LDAP_PRINCIPAL and LDAP _PASSWORD environment variables. These environment variables are set using the env property. Their contents are referenced from a `Secret` that you created in a previous step.

  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.

  For more information regarding surveyorConfig.quarkus.* properties, see Using Security with an LDAP Realm in the Quarkus documentation.

  Note: Groups are not required for authentication. You might want to configure the surveyorConfig.quarkus.security.ldap.identity-mapping.attribute-mappings.* properties accordingly.

- 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.

6. Apply your configuration.

   The helm upgrade command you run will differ depending on whether FIPS mode is enabled or not. If FIPS mode is enabled, you must include the authentication key file using the --set-file option. Otherwise, omit this option to use automatic key generation.

   For Non FIPS

   ```
   helm upgrade CLOUDERA-SURVEYOR [***CHART***] \
     --namespace [***NAMESPACE***] \
     --values [***VALUES.YML***]
   ```

   For FIPS

   ```
   helm upgrade CLOUDERA-SURVEYOR [***CHART***] \
     --namespace [***NAMESPACE***] \
     --values [***VALUES.YML***] \
     --set-file surveyorConfig.surveyor.authentication.keys.active=[***PATH
    TO AUTHENTICATION KEY FILE***]
   ```

Results

LDAP authentication is enabled. Users are required to authenticate to access the Cloudera Surveyor UI.

Related Information

Using Security with an LDAP Realm | Quarkus

Updating the authentication key

Learn how to update an authentication key that Cloudera Surveyor uses to generate authentication tokens. Cloudera recommends you update the key regularly. Alternatively, you might need to update the key as a security measure to log out all users.

About this task

You update the authentication key by updating surveyorConfig.surveyor.authentication.keys.active with a new key that you generate. Additionally, you temporarily set surveyorConfig.surveyor.authentication.keys.passive to the old key. Having both keys active at the same time is required to ensure a rolling transition of keys and prevents users from being forced to relogin multiple times. After the new key is added and active, you remove the old key to deactivate it.

Before you begin

A complete Cloudera Surveyor configuration (a full values.yaml file with all configured properties) is required to complete this task. Ensure you have the values file ready. You can retrieve all values for your installation using the `helm get values command.

```
helm get values CLOUDERA-SURVEYOR \
  --namespace [***NAMESPACE***] \
  --all
```

Procedure

1. Generate a new 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.
2. Activate your new key.
   a) Extract your old key.

   ```
   kubectl get secret cloudera-surveyor-auth-keys \
     --namespace [***NAMESPACE***] \
     -ojsonpath='{.data.key-active}' \
   | base64 --decode > old-key
   ```

   b) Activate the new key.

   ```
   helm upgrade CLOUDERA-SURVEYOR [***CHART***] \
     --namespace [***NAMESPACE***] \
     --values [***VALUES.YAML***] \
     --set-file surveyorConfig.surveyor.authentication.keys.active=[***PATH
    TO NEW KEY FILE***] \
     --set-file surveyorConfig.surveyor.authentication.keys.passive=old-key
   ```

   c) Rolling restart Cloudera Surveyor.

   ```
   kubectl rollout restart deployment/CLOUDERA-SURVEYOR \
   --namespace [***NAMESPACE***]
   ```

   d) Wait until the deployment is successfully restarted.

   Monitor the restart with the kubectl rollout.

   ```
   kubectl rollout status deployment/CLOUDERA-SURVEYOR \
     --namespace [***NAMESPACE***] \
   ```

```
  --watch
```

The restart is successful once you see the following message.

```
deployment "cloudera-surveyor" successfully rolled out
```

e) Delete the old key file you created in a previous step.

```
rm old key
```

3. Deactivate your old key by removing it from configuration.

```
helm upgrade CLOUDERA-SURVEYOR [***CHART***] \
  --namespace [***NAMESPACE***]
  --values [***VALUES.YAML***] \
  --set-file surveyorConfig.surveyor.authentication.keys.active=[***NEW
 KEY FILE***]
```

Notice that surveyorConfig.surveyor.authentication.keys.passive (the old key) is not provided. This removes and deactivates the old key.

4. Rolling restart Cloudera Surveyor.

```
kubectl rollout restart deployment/CLOUDERA-SURVEYOR \
  --namespace [***NAMESPACE***]
```

5. Wait until the deployment is successfully restarted.

Monitor the restart with the kubectl rollout.

```
kubectl rollout status deployment/CLOUDERA-SURVEYOR \
  --namespace [***NAMESPACE***] \
  --watch
```

The restart is successful once you see the following message.

```
deployment "cloudera-surveyor" successfully rolled out
```

Results

The new authentication key is added. Authentication tokens are now signed with the new key. The old key is no longer accepted. All users are required to relogin.

What to do next

The new authentication key is sensitive data. Cloudera recommends that you delete the new key file you generated.

Authorization in Cloudera Surveyor for Apache Kafka

Cloudera Surveyor ensures secure access and permission management through Kafka Access Control Lists (ACLs). Cloudera Surveyor does not implement its own authorization model, and fully relies on the Kafka ACLs. Additionally, Cloudera Surveyor supports a two-level principal mapper model. This means you can configure a centralized and a per-cluster principal mapping rule.

Requirements

- The Kafka cluster's authorizer implementation must support the `DESCRIBE_ACLS` Kafka API.

How authorization works

When authorization is enabled for a Kafka cluster in Cloudera Surveyor configuration, Cloudera Surveyor periodically fetches ACLs from Kafka, and enforces them on all endpoints. This means that all information shown by Cloudera Surveyor follows the same authorization rules as in Kafka.

For example, querying the min.isr property of a Kafka topic requires a TOPIC DESCRIBE_CONFIGS permission, which means that Cloudera Surveyor will only provide the min.isr if the authenticated user has the exact permission as described by the Kafka ACLs.

The only deviation from the Kafka ACL model in Cloudera Surveyor is how clusters are shown to users on the Cloudera Surveyor API. A user needs the CLUSTER DESCRIBE permission to access a cluster registered in Cloudera Surveyor.

Principal mapping rules

It is possible that authentication configured in Cloudera Surveyor produces different principal names than the ones configured in the integrated Kafka clusters.

To match the users logged into Cloudera Surveyor with the principals configured in Kafka, you can use the following two principal mapping rules:

- **A global mapping rule** – configured with surveyorConfig.surveyor.authentication.principalMappingRule
- **A per-cluster mapping rule** – configured with clusterConfigs.clusters[*].authorization.principalMappingRule

Both of these default to the identity mapping (DEFAULT). When the global mapping rule is configured, it applies to all principals in all clusters. When the per-cluster mapping rule is configured, it only applies to the specific cluster, and it is applied after the global mapping rule. The mapping rules follow the same pattern as used by the Kafka ssl.principal.mapping.rules property.

Table 1: Principal mapping rule example

Authenticated user	Global mapping	Per-cluster mapping	Mapped principal
user_john_smith	DEFAULT	DEFAULT	user_john_smith
user_john_smith	RULE:^user_(.*)$/u_$1/	DEFAULT	u_john_smith
user_john_smith	DEFAULT	RULE:^(.*)_(.*)_(.*)$/\$1_\$2/	user_john
user_john_smith	RULE:^user_(.*)$/u_$1/	RULE:^(.*)_(.*)_(.*)$/\$1_\$2/	u_john

Related Information

Operations and Resources on Protocols | Apache Kafka
Customizing SSL User Name | Apache Kafka
ssl.principal.mapping.rules | Apache Kafka